CS 6364 Knowledge Representation

Rinkle Seth
Department of Computer Science
The University of Texas at Dallas Richardson, TX 75080 USA
rcs170004@utdallas.edu

***Abstract*— Knowledge representation is the way we represent the information which we know about a world using assertions and first order logic. There are classes, objects, properties of objects, relations between classes, etc. which depict the way these classes are related to each other. For example, a dog is an animal so that would be a subclass of animal inheriting properties of animal class like walking, eating, etc. This makes it easier to represent information and make a computer understand how it works. The proposed paper focuses on knowledge representation for a supermarket and design of a question-answering system using Prover9 which answers queries given in logic form.**

***Keywords*— Axioms, knowledge representation, first order logic**

I. INTRODUCTION

Knowledge representation is widely used today to represent information in a way that a computer can understand and utilize to solve complicated tasks such as healthcare diagnosis systems. In general, we can represent information of a world using classes, objects, relations, etc. Classes are used to refer to a group of objects/instances exhibiting same properties. E.g. Shop.3. All shops have properties like selling, buying, etc.

These classes can also have subset classes. For example, shop can have supermarket as a subset class since supermarket refers to a more specific class exhibiting properties similar to a shop but has more specificity like selling only dairy products, personal items, food items. Basically, the scope would be limited as compared to a broad range of a shop. Constants are specific instances of classes. For example, John represents a specific instance of class person, so it is a constant. Similarly, if I am referring to a specific place like North Berkeley or a specific store like Safeway, it will be treated as a constant.

This paper describes a question answering system based on a supermarket's knowledge base.

II. KNOWLEDGE BASE DESCRIPTION AND DESIGN

The knowledge base captures the general structure of a supermarket as well as some modifications according to the scope of problems 12.5 and 12.6. The following assertions are then converted into first order logic i.e. we turn them into axioms/logic form so that a computer can make sense of them.

The knowledge captured in the domain is as follows:

- Starting with the scenario given below, I made assertions to **capture** it.

Yesterday John went to the North Berkeley Safeway supermarket and bought two pounds of tomatoes and a pound of ground beef.

1. T represents the instance of tomato which John bought.
2. t represents the time at which the purchase was made.
3. B represents the instance of beef which John bought.
4. Asserted that John is a person.
5. John bought tomatoes at time t from Safeway.
6. Basically, captured the scenario.

- I made assertions to capture the **generic scenario** of a supermarket.
  1. For example, a supermarket is a shop.
  2. The kind of items it sells include dairy products, personal items, food items, etc.
  3. Customers/employees can scan the items.
  4. Customers get a receipt after buying an item.
  5. Supermarkets have employees and customers.
  6. Customers can use carts for shopping, stores provide online delivery, etc.
  7. Employees restock items and assist customers, etc.

- **Constants** used in the knowledge base are as follows:
  1. B - Beef
  2. T - Tomato
  3. t - time
  4. D - Deodorant

5. Cart - Cart
6. Safeway
7. NBS
8. NorthBerkeley
9. John
10. Mary
11. Pound, Ounce, Kilogram
12. G - Gas

- **Predicates** include the following:
  1. Binary predicates : sells, use, deliver, get, sellItems, has, make, weight, owns, eats, etc.
  2. Unary predicates: gas, meat, fooditem, supermarket, shop, item, item, beverage, customer, person, adult, employee, etc.
  3. Others: bought, lessMoney, moreMoney, scan, assist, restock, brings, etc.

- Assertions for making the domain even **more generic.**
  1. For example, supermarkets sell food items.
  2. Vegetables are food items.
  3. Broccoli, Tomato, etc. are vegetables.
  4. Meat is a food item.
  5. Fruits like mango, pineapple are food items, etc.
  6. Personal items include deodorant, soap, bodywash, etc.
  7. Beverages include tea, coffee, alcohol, soda, etc.
  8. All of these are items which could be sold in the market.

- **Assertions** for answering questions in problems **12.5 and 12.6**.
  1. For example, Shell station is a shop that sells gas.
  2. People buy food items to eat.
  3. Assumed that customers who buy items from shops are adults.
  4. Customers have less money after buying items and shops have more money after selling items.
  5. Customers bring money to purchase items.
  6. If people buy items from the same store at the same time, then they see each other.
  7. Supermarkets do not make items.
  8. Shops own items before they are sold and customers own them after purchase, etc.

III. First order logic

The next step is converting the assertions made in step II into logic form i.e. first order logic. First order logic has quantifiers like for all and there exists to denote all values, or some value is true in a given sentence.
I won't go into the details of converting it into first order logic in this section. But, let me just give an example for the following sentence into first order logic.
Shop sells items.
all x (exists y shop(x) & item(y) -> sells (x, y))
Here the scope of shop is for all because it's true for all shops, all shops sell items. But scope of items is exists since all shops do not sell all items.
Supermarkets do not sell furniture.

IV. Assumptions

- Supermarkets sell **only food items, personal items, dairy products**. Have limited scope to only the aforementioned items; not included clothes, furniture, etc.
- Assumed that a pound has **5 tomatoes**.
- If a customer x makes a purchase of item y from store z at time t then the customer **pays at time t**.
- Customers at supermarkets **use carts**.

V. Sample Question Asked to the system. (problems 12.5 and 12.6)

1. Did John buy any meat?

```
exists x exists z exists t(meat(x) &
bought(John, x, z, t)).

============================== PROOF
================================
% Proof 1 at 0.02 (+ 0.00) seconds.
% Length of proof is 9.
% Level of proof is 4.
% Maximum clause weight is 5.000.
% Given clauses 0.

17 (all x (beef(x) -> meat(x))) #
label(non_clause).  [assumption].
79 (exists x exists z exists t (meat(x) &
bought(John,x,z,t))) # label(non_clause)
# label(goal).  [goal].
159 -beef(x) | meat(x).  [clausify(17)].
160 beef(B).  [assumption].
165 -meat(x) | -bought(John,x,y,z).
[deny(79)].
166 meat(B).  [resolve(159,a,160,a)].
300 bought(John,B,Safeway,t).
[assumption].
303 -bought(John,B,x,y).
[resolve(166,a,165,a)].
304 $F.  [resolve(303,a,300,a)].
============================== end of
proof ==========================
```

2. If Mary was buying tomatoes at the same time as John, did he see her?

```
bought(Mary, T, Safeway, t)-> see(John,
Mary)|see(Mary, John).

============================== PROOF
================================

% Proof 1 at 0.02 (+ 0.00) seconds.
% Length of proof is 16.
% Level of proof is 4.
% Maximum clause weight is 20.000.
% Given clauses 8.

4 (all x ((exists y all z exists u
customer(u)) & supermarket(x) &
employee(y) & item(z) & restock(y,z,x) &
assist(y,u))) # label(non_clause).
[assumption].
13 (all x ((exists y shop(x)) & has(x,y)
& employee(y) & customer(y))) #
label(non_clause).  [assumption].
37 (all x ((all y exists u exists z
person(x)) & person(y) & item(u) &
shop(z) & bought(x,u,z,t) &
bought(y,u,z,t) -> see(x,y) | see(y,x)))
# label(non_clause).  [assumption].
79 bought(Mary,T,Safeway,t) ->
see(John,Mary) | see(Mary,John) #
label(non_clause) # label(goal).  [goal].
114 item(x).  [clausify(4)].
118 -person(x) | -person(y) | -item(z) |
-shop(u) | -bought(x,z,u,t) | -
bought(y,z,u,t) | see(x,y) | see(y,x).
[clausify(37)].
135 shop(x).  [clausify(13)].
142 -person(x) | -person(y) | -shop(z) |
-bought(x,u,z,t) | -bought(y,u,z,t) |
see(x,y) | see(y,x).
[resolve(118,c,114,a)].
304 person(John).  [assumption].
305 bought(John,T,Safeway,t).
[assumption].
307 person(Mary).  [assumption].
308 bought(Mary,T,Safeway,t).
[deny(79)].
309 -see(John,Mary).  [deny(79)].
310 -see(Mary,John).  [deny(79)].
311 -person(x) | -person(y) | -
bought(x,z,u,t) | -bought(y,z,u,t) |
see(x,y) | see(y,x).
[resolve(142,c,135,a)].
316 $F.
[ur(311,b,307,a,c,305,a,d,308,a,e,309,a,f
,310,a),unit_del(a,304)].

============================== end of
proof ==========================
```

3. Are there other people in Safeway while John is there?

```
exists x (has(Safeway, John) &
has(Safeway, x) & employee(x)).

============================== PROOF
================================

% Proof 1 at 0.02 (+ 0.00) seconds.
% Length of proof is 8.
% Level of proof is 3.
% Maximum clause weight is 3.000.
% Given clauses 0.

4 (all x ((exists y all z exists u
customer(u)) & supermarket(x) &
employee(y) & item(z) & restock(y,z,x) &
assist(y,u))) # label(non_clause).
[assumption].
13 (all x ((exists y shop(x)) & has(x,y)
& employee(y) & customer(y))) #
label(non_clause).  [assumption].
79 (exists x (has(Safeway,John) &
has(Safeway,x) & employee(x))) #
label(non_clause) # label(goal).  [goal].
109 -has(Safeway,John) | -has(Safeway,x)
| -employee(x).  [deny(79)].
110 employee(x).  [clausify(4)].
305 has(x,y).  [clausify(13)].
310 -has(Safeway,John) | -has(Safeway,x).
[resolve(109,c,110,a)].
311 $F.
[copy(310),unit_del(a,305),unit_del(b,305
)].

============================== end of
proof ==========================
```

VI. TEST QUESTIONS AND OUTPUT SCREENSHOTS:

I am specifying some of the questions which I asked to the system and which were answered correctly. These aren't questions from problems 12.5 and 12.6 but some of my own set of test questions.

1. Is John a customer?

   ```
   customer(John).
   ```

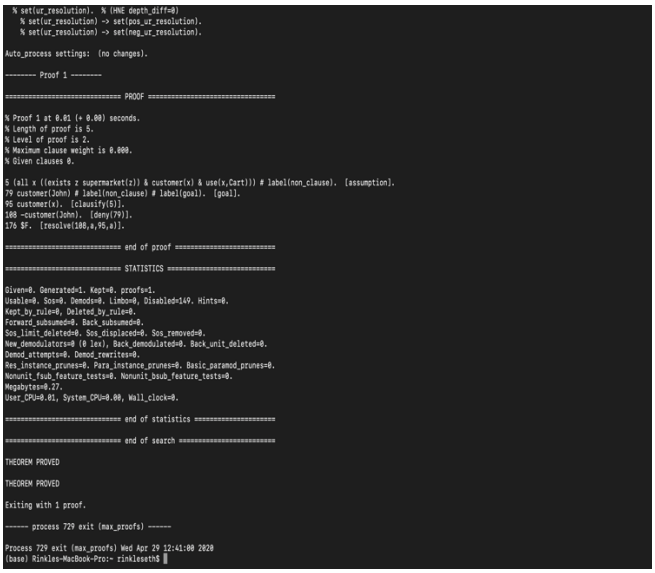

Fig.1 - Prover9 output for John is a customer

2. Does Safeway sell pork?

```
all x (pork(x) -> sells(Safeway, x)).
```

Fig.2 - Prover9 output for Safeway sells pork

3. Did John get a receipt?

```
get(John, receipt).
```

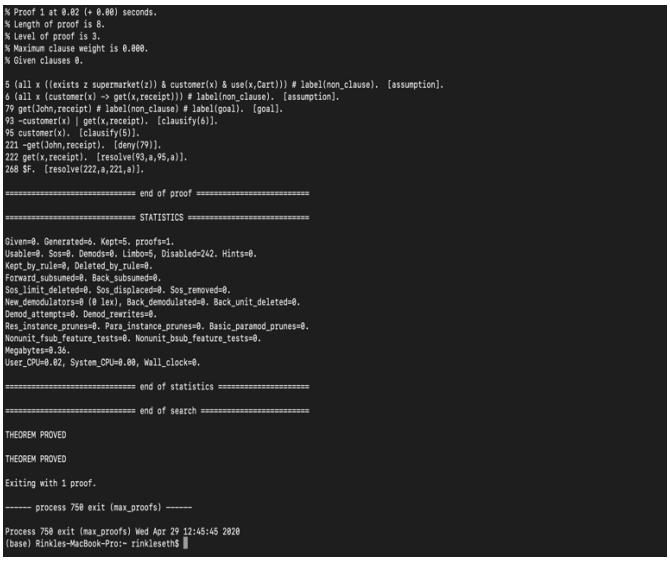

Fig.3 - Prover9 output for John got a receipt.

4. Does Safeway deliver items?

```
exists x (item(x) &
deliver(Safeway, x)).
```

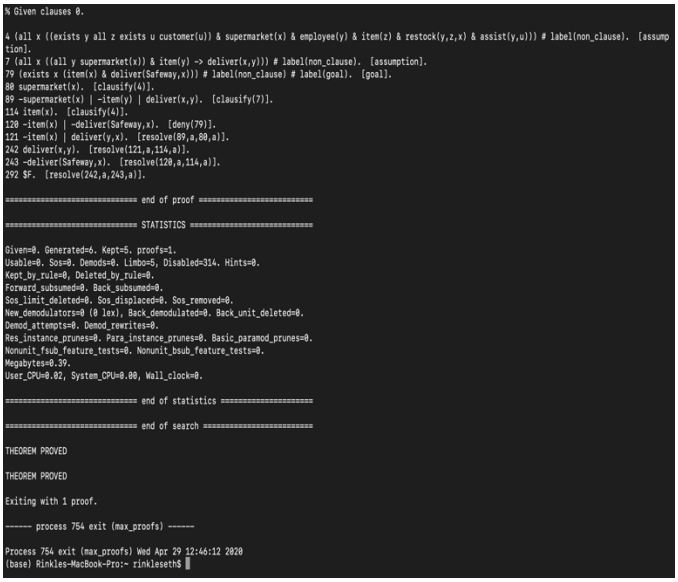

Fig.4 - Prover9 output for Safeway delivers items.

5. Did John use cart?

```
use(John, Cart).
```

Fig.5 - Prover9 output for John uses cart

VII. PERFORMANCE OF TOOL:

A question answering system for supermarkets was successfully designed. It currently has a consistent set of axioms which can answer a generic set of questions. Also, it's able to make inferences quite nicely. It can inherit properties of super classes and infer rules.

The performance of the tool was tested on the test questions mentioned in Section VI, all of them were answered correctly. It was tested on the sample questions mentioned in problems 12.5[1] and 12.6[1] as well and the system was able to correctly answer 12/15 questions. The remaining three questions required numeric comparisons and arithmetic calculations which is currently not supported in Prover9.

VIII. CONCLUSION:

Knowledge base systems can be widely used in different domains like healthcare, education, grocery stores, etc. The domain experts can be approached for information on what nodes the system should contain, how are they related to each other, what are the classes and objects, etc.

It makes inference possible which is one of the important things in AI. When systems start inferring the current knowledge and add new rules according to what they infer, they would then be referred as intelligent. It also makes human tasks easier that way. In medical diagnosis systems, it could help doctors in diagnosis of patients, it could suggest what all diseases could be possible given the symptoms.

In this paper, I have just demonstrated a small example of what it would be like to have such a tool. It intelligently understands information and infers knowledge based on that which can be used for further queries.

Though it's extremely difficult to design a system which answers all questions correctly, but it still does a good job.

REFERENCES

[1] Stuart Russell and Peter Norvig. Artificial Intelligence: A Modern Approach. Prentice Hall Press, 2009

[2] Christiane Fellbaum. WordNet: An Electronic Lexical Database. Bradford Books, 1998.

[3] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In Proceedings of the 2008 ACM SIGMOD international conference on Management of data, pages 1247–1250, 2008.

[4] Wordnet – A Lexical Database

[5] FreeBase