Introduction to AI - assignment B1

Oded Yechiel

1/1/19

Exercise 1 Introduction

In this assignment a theorem proving resolution program was developed. The program gets as an input a file containing axioms in Conjunctive Normal Form (CNF) and a query (also in CNF) that is desired to prove.

The program output is one of the 3:

- *Query is False*
- *CONTRADICTION Query is True*
- *Query can not be disproven*

The program is written in `Python` and can be found in `https://github.com/odedyec/IntroToAI/tree/master/assignmentB1`.

Exercise 2 Conventions

The parser of the program can parse very neatly text into understandable sentences by the solver engine. In order to conserve time of development there are several strict conventions which are have to be met:

1. Each line is an axiom and can contain as many fluents or predicats as desired. From hereon, we will consider all of the fluents as predicats.
2. It is recommended to start the fluents / predicats with a capital letter. For example, Enemy(...), `Weapon(...)`, `Loc(...)`.
3. Predicats are separated by a **V** sign. It is disallowed to use **V** in any variable or predicat.
4. A predicat has to have at least one variable.
5. A negated predicat starts with a "!" sign.
6. A variable is a Skolem (constant) iff the variable starts with a capital letter. For example, `Criminal(West)` - West is a Skolem variable.
7. Adding between variables is possible iff the variable is defined as an integer. For example, `Carrying(0, S0)` - the first variable is an integer and it is possible to add on this variable.
8. Variables that are lower case can be assigned by the program.

9. There are some protected names for predicats that have a unique definition.

   - Result(a, s, s1) - this predicat gets as an input `s`, as skolem value (e.g. S0) and *s1* as a skolem value with the next following number (e.g. S1).
   - Plus(a, b, c) - this predicat gets as an input two integer valued defined variables, a and b, and sets the variable c as the sum of them.
   - Greater(a, b, s) - this predicat checks if $a > b$, if so it creates the predicat $\neg$ `Timeup(s)` otherwise, `Timeup(s)`

   These predicats disappear after they are called, as opposed to regular predicats that disappear only when combined with their complementary predicat.

Exercise 3 Code explanation

There are mainly three objects which are the building blocks of the program: Variable, Predicats and Axioms.

3.1 Variables

The variable object is a mutable object, which means that it is not copied. Therefore, if several predicats are pointing to the same variable and the variable changes, all of the variables change as a result.

A variable has to properties: a value (default: None) and a symbol. A variable with no value is shown by its symbol and its value can be set. A value that is set can not be changed, unless its value is an integer, and is shown by its value.

3.2 Predicats

The predicat object has a name defining it and a list of vars. It also has a boolean variable stating if its negated or not.

The object has also two methods: `substitute` and `is_complement`. The substitue method should get a list of variables objects. These variables are compared with the predicats own variables, and for each variable one of the three can occur:

- Both of the variables are not defined, in which case the variables are combined to a new variable.
- One of the variables is defined and one is not. In this case the undefined variable is defined with the value of the second.
- Both of the variables are defined. In this case, if the variables have the same value the program continues, otherwise we have a contradiction and the query is true.

3.3 Axiom

The axiom object contains a list of predicats.

The axiom object performs the unification with another axiom or predicat. The unification with a predicat is performed by checking if it is complement with any of the predicats contained by the axiom. If so, substitution is performed between the variables, and these predicats are removed from the axiom. If the predicat is not complement with any of the predicats of the axiom, the predicat is added to the axiom.

3.4 The knowledgeBase object and main

The knowledge base is an extension to the Axiom object. The KnowledgeBase has also a useful printing method and a resolve full list for automatic resolution of a list with axioms.

The main file simply parses a text file and runs the Knowledge base until the knowledge base is empty, or there is no more resolutions to be performed. Each problem set will have its own main file.

Exercise 4 Examples

4.1 Criminal West

The very well known query if captain west is a criminal is performed by negation using the following query ¬`Criminal(West)`. The output is seen in Fig. 1.

```
_____
Resolving  Missile(x2) V Weapon(x2) With  Criminal(West)
Result  Criminal(West) V  Missile(x2) V Weapon(x2)
_____-
Resolving  American(x) V  Weapon(y) V  Sells(x,y,z) V  Hostile(z) V Criminal(x) With  Criminal(West) V
Missile(x2) V Weapon(x2)
Result  Missile(y) V  American(West) V  Sells(West,y,z) V  Hostile(z)
_____-
Resolving American(West) With  Missile(y) V  American(West) V  Sells(West,y,z) V  Hostile(z)
Result  Missile(y) V  Sells(West,y,z) V  Hostile(z)
_____-
Resolving Missile(M1) With  Missile(y) V  Sells(West,y,z) V  Hostile(z)
Result  Sells(West,M1,z) V  Hostile(z)
_____-
Resolving  Missile(x3) V  Owns(Nono,x3) V Sells(West,x3,Nono) With  Sells(West,M1,z) V  Hostile(z)
Result  Hostile(Nono) V  Missile(M1) V  Owns(Nono,M1)
_____-
Resolving Missile(M1) With  Hostile(Nono) V  Missile(M1) V  Owns(Nono,M1)
Result  Hostile(Nono) V  Owns(Nono,M1)
_____-
Resolving Owns(Nono,M1) With  Hostile(Nono) V  Owns(Nono,M1)
Result  Hostile(Nono)
_____
Resolving  Enemy(x4,America) V Hostile(x4) With  Hostile(Nono)
Result  Enemy(Nono,America)
_____-
Resolving Enemy(Nono,America) With  Enemy(Nono,America)
Result
==============
Query is False
==============
```

Figure 1: Output of the program.

4.2 Hurricane environment

The hurricane environment is explained in length in assignment 3 (also can be found in the git repo).

Here I have provided a list of axioms that the user can choose which one to apply and at which time. The menu the user sees is

— What would you like to reolve —
0. No way to coninue?
1. Weight(E1,3) V Edge(E1,K0,K1) V Traverse(E1)
2. Weight(E2,1) V Edge(E2,K1,K2) V Traverse(E2)
3. Weight(E3,1) V Edge(E3,K2,K3) V Traverse(E3)
4. Edge(e,k1,k2) V Edge(e,k2,k1)
5. Traverse(a) V Carrying(x,s) V Carrying(x,s1) V Weight(a,w) V Time(t,s) V Time(t2,s1) V Edge(a,v1,v2) V Loc(v1,s) V Loc(v2,s1) V Result(a,s,s1) V Plus(t,w,t2)
6. At(v2,p) V Carrying(x,s1) V Carrying(y,s1) V At(v2,0) V Plus(x,p,y)
7. Shelter(k) V Carrying(x,s) V Carrying(0,s)
8. Time(t,s) V Deadline(t2) V Greater(t,t2,s)
9. At(K3,2)
10. Shelter(K0)
11. Carrying(0,S6) V Loc(K0,S6)
12. Custom

Figure 2: Hurricane environment axiom user menu

The knowledge base is started with the following information:

Time(0, S0)
Carrying(0, S0)
Loc(K0, S0)
Deadline(11)

Figure 3: Initial info for the knowledge base

The query of the knowledge base is ¬`Timeup(S6)` (will time elapse) given the plan `Traverse(E1) V Traverse(E2) V Traverse(E3) V Traverse(E3) V Traverse(E2) V Traverse(E1)`.

Entering a list of axioms of $[1, 5, 2, 5, 3, 5, 9, 6]$ will bring the agent to Vertex 3 and will pick-up the people there (`Loc(K3, S3) V Carrying(2, S3)`).

Now we wish to discard the `At(K3, 0)` predicat, and go back to Vertex 0. For that we will need to use the undirected property of the graph using axiom 4. Thus the input list will be $[3, 4, 5, 2, 4, 5, 1, 4, 5, 10, 7]$.

Discarding the position and carrying predicats using axiom 11 and applying axiom 8 will result in an empty knowledge base, thus proving there will be no sufficient time to rescue the two people given the plan.

The output of the program is provided in the text file in the repo.