Jerry Chienyu Chen
November20/2016

CMPT310 Assignment 3 Report

Discussion About Algorithm

At the beginning of implementing this assignment question, I tried to go over the given KB on the assignment by hand on paper using the backward chaining algorithm given on the assignment. At first, the terms rule, goals, atom, fact, body and head were quite confusing while I was reading through the assignment, but as I go through them carefully, everything starts to make sense with the given pseudocode. Basically the main idea of this assignment is about backward chaining. Backward chaining is simply an algorithm that does the same as what human does when looking at a knowledge base. Let's take a look at the example 2 given and elaborate the backward chaining algorithm on the way

Example 2:

The rules: p, $p \Rightarrow q$, $p \wedge q \Rightarrow r$, $s \Rightarrow r$ would be expressed: p, $q\ p$, $r\ p\ q$, $r\ s$

Say if we have a query r for this example. As human beings, we would look into what atoms make r becomes true at first and check if the atoms of r are all true. If yes, then this implies that r is true. So at first if we scan through the KB top down, and we look into p∧q=>r. If p and q are both true in this rule, then r has to be true too. To confirm that, we then look into p and see if p is true. Scanning through the KB top down again, we can see that p is listed as a fact on the first line of the KB which means that p is true. Afterwards, we trace back to p∧q=>r. since p is clarified to be true already, then we check the next atom which is q and see if q is true. Looking into q, we see the second rule showing as p=>q which tells us that p implies q. We know intuitively that q is also true since p is true reasoning p implies q. Since we proved that both atoms p and q are true within the KB, this implies that r is true in the KB as well.

To generalize this process, we repeat the above procedure over and over again looking through all the rules in the KB with the head value the same as the query we are looking for. If one of the rules turns out to imply that the query is true, then the process ends and then return true saying that this KB is true with demanded query. On the other hand, if none the rules in the KB with the head value the same as the query turns out to imply that the query is true after scanning through the entire KB, then we know that the query is false. Hence, the KB is false with the demanded query.

Generalizing this repeated procedure into an algorithm, this is exactly what the backward chaining algorithm does along with various data structures storing the rules and goals.

Jerry Chienyu Chen
November20/2016

Backward Chaining Algorithm Implementation

The core of this assignment is implemented with the backward chaining algorithm. Basically how I did it was simply turn the given pseudocode from the assignment into working Python code showing as below.

solve(goals):
    if $goals = ()$ then return true
    let $a = first(goals)$; $goals = rest(goals)$
    for each $r \in rules$ where $head(r) = a$
        if $solve(append(body(r), goals)) = \text{true}$
            return(true)
    return(fail)

The parameter goals taking into the solve method is a Python list of atoms that we are tracking. If the goals list is empty, then we return true. If not, then then we do the rest showing on the pseudocode where you have a variable being assigned with the first value of the goals list. Here, mine was named as “firstGoal”. And then we want to update the goals list after firstGoal has been taken out for search among the rules. Afterwards, we run a for loop to iterate and scan through all the rules in the knowledge base that has its rule head the same value as the firstGoal, and that specific rule is what we are looking for. We then append the body values of that rule into the goals list, and then run the appended goals into the backward chaining method again using recursion. The abstract idea of recursion at this section is elaborated at the previous section at the discussion about algorithm.

```
def backwardChaining(self):
    global goals
    # if goals=NULL then return true
    print("goals:", ','.join(goals))
    if not goals:
        print("goal is NULL")
        print("This KB is ", end="")
        self.result = True
        return self.result

    # let a=first(goals); goals=rest(goals)
    firstGoal = goals[0]
    print("Current searching head:", firstGoal)
    goals = goals[1:]

    if goals:
        print("Rest of goals:", ','.join(goals))
    elif not goals:
        print("Rest of goals: NULL")

    # for each r belongs to rules where head(r)=a
    for rule in self.rulesList:
        # rule in this case is each list within the rulesList
        if (rule[0] == firstGoal):
            # if the head of the rule = firstGoal
            # then we append that specific rule's body to goals
            print("Chosen rule we are looking at right now: ", end="")

            if (len(rule) == 1):
                print(''.join(rule))
                print("Message:", ''.join(rule), "is a FACT")
                print("Rule body: NULL")
            else:
                print('∧'.join(rule[1:]), "=>", rule[0])
                print("Rule body:", '∧'.join(rule[1:]))

            print("Append these body atoms into goals...")
            goals.append(rule[1:])
            goals = [inner for outer in goals for inner in outer]
            print("goals after append:", ','.join(goals))
            print("")

            # and then we do the recursion
            if (self.backwardChaining()):
                self.result = True
                return self.result
                # return True

    # else the KB is false
    self.result = False
    return self.result
    # return False
```

Jerry Chienyu Chen
November20/2016

Output Elaboration

I created a short KB here just to demonstrate how my result output works. At first, as we enter into the core backward chaining method, it displays the query of the KB at the first line as “Query: r”. And then we start looking for query r among the rules, and we find a chosen rule. We display the current chosen rule p∧q=>r out on the screen just to let the user know which rule we are looking into right now at this moment. Now we can see that the atoms of head r are p and q, so then we append them into goals list as shown at the last line of the first chunk of the output as “goals after append: p∧q”. Now we enter into another iteration, and then we look for the first one in the goals list which is p in this case. So then we tell the user that “Current searching head” we are searching for is p. Again we tell the user that which rule is chosen and display the information on the screen. Showing on the second chunk of the output, you can see that the chosen rule p is one single atom that is not implying to anything else not being implied by other atoms. So we know right away that p is a fact. Hence, we display this message to the screen saying “Message: p is a FACT” so we know that p is TRUE. Afterwards, we do the same thing looking for q and find out that q is a fact as well. So we display “Message: q is a FACT” showing that q is TRUE as well. Since both atoms of query r are TRUE, then this implies that query r is TRUE too. Hence, we display “This KB is True” at the very end and terminates. This goes on iteration by iteration the same way as described above. The output is quite clear step by step the same way as how us human beings look at a KB with given query.

KB:

```
r
p
q
r p q
```

Result Output:

```
Query: r
Adding query into goals...

goals: r
Current searching head: r
Rest of goals: NULL
Chosen rule we are looking at right now: p∧q => r
Rule body: p∧q
Append these body atoms into goals...
goals after append: p,q

goals: p,q
Current searching head: p
Rest of goals: q
Chosen rule we are looking at right now: p
Message: p is a FACT
Rule body: NULL
Append these body atoms into goals...
goals after append: q

goals: q
Current searching head: q
Rest of goals: NULL
Chosen rule we are looking at right now: q
Message: q is a FACT
Rule body: NULL
Append these body atoms into goals...
goals after append:

goals:
goal is NULL
This KB is True

Process finished with exit code 0
```

Jerry Chienyu Chen
November20/2016

Testing with Given Data Sets

Example 1:

KB:

```
a
p
q p
d
g q j
e f
c f d
c d g
c e d
b j
a b c
j
```

Result Output:

```
Query: a
Adding query into goals...

goals: a
Current searching head: a
Rest of goals: NULL
Chosen rule we are looking at right now: b∧c => a
Rule body: b∧c
Append these body atoms into goals...
goals after append: b,c

goals: b,c
Current searching head: b
Rest of goals: c
Chosen rule we are looking at right now: j => b
Rule body: j
Append these body atoms into goals...
goals after append: c,j

goals: c,j
Current searching head: c
Rest of goals: j
Chosen rule we are looking at right now: f∧d => c
Rule body: f∧d
Append these body atoms into goals...
goals after append: j,f,d

goals: j,f,d
Current searching head: j
Rest of goals: f,d
Chosen rule we are looking at right now: j
Message: j is a FACT
Rule body: NULL
Append these body atoms into goals...
goals after append: f,d

goals: f,d
Current searching head: f
Rest of goals: d
Chosen rule we are looking at right now: d∧g => c
Rule body: d∧g
Append these body atoms into goals...
goals after append: d,d,g

goals: d,d,g
Current searching head: d
Rest of goals: d,g
Chosen rule we are looking at right now: d
Message: d is a FACT
Rule body: NULL
Append these body atoms into goals...
goals after append: d,g

goals: d,g
Current searching head: d
Rest of goals: g
Chosen rule we are looking at right now: d
Message: d is a FACT
Rule body: NULL
Append these body atoms into goals...
goals after append: g

goals: g
Current searching head: g
Rest of goals: NULL
Chosen rule we are looking at right now: q∧j => g
Rule body: q∧j
Append these body atoms into goals...
goals after append: q,j

goals: q,j
Current searching head: q
Rest of goals: j
Chosen rule we are looking at right now: p => q
Rule body: p
Append these body atoms into goals...
goals after append: j,p

goals: j,p
Current searching head: j
Rest of goals: p
Chosen rule we are looking at right now: j
Message: j is a FACT
Rule body: NULL
Append these body atoms into goals...
goals after append: p

goals: p
Current searching head: p
Rest of goals: NULL
Chosen rule we are looking at right now: p
Message: p is a FACT
Rule body: NULL
Append these body atoms into goals...
goals after append:

goals:
goal is NULL
This KB is True

Process finished with exit code 0
```

Jerry Chienyu Chen
November20/2016

Example 2:

KB:

```
r
p
q p
r p q
r s
```

Result Output:

```
Query: r
Adding query into goals...

goals: r
Current searching head: r
Rest of goals: NULL
Chosen rule we are looking at right now: p∧q => r
Rule body: p∧q
Append these body atoms into goals...
goals after append: p,q

goals: p,q
Current searching head: p
Rest of goals: q
Chosen rule we are looking at right now: p
Message: p is a FACT
Rule body: NULL
Append these body atoms into goals...
goals after append: q

goals: q
Current searching head: q
Rest of goals: NULL
Chosen rule we are looking at right now: p => q
Rule body: p
Append these body atoms into goals...
goals after append: p

goals: p
Current searching head: p
Rest of goals: NULL
Chosen rule we are looking at right now: p
Message: p is a FACT
Rule body: NULL
Append these body atoms into goals...
goals after append:

goals:
goal is NULL
This KB is True

Process finished with exit code 0
```