# Mandatory assignment Week 6

Made by Christian Bank Lauridsen chbl@itu.dk.
Worked together with Lucas Frey Torres Hanson luha@itu.dk

## Mandatory exercise

## Part 1

In which situations can it be an advantage to do backward-chaining rather than forward-chaining?

To answer the question, we need to understand the differences between forward and backward chaning. `Forward chaining` determines if a query is entailed by a knowledge base of definite clauses. It starts with a set of known data from the knowledge base and uses inferece rules to add new conclusion to the set of known data. This process continues until the query is added is added to the knowledge base. `Backward chaning` works backwards from a query. Instead of trying to find new conclusion by already known facts, it tries to find the implications that led to the conclusion (the query).

`Backward chaining` is therefore very usefull when answering specific questions, because it only explores the parts of the knowlegde base that are relevant to a query. In comparision to `forward chaning`, `forward chaining` could explore larger parts of the knowledge base that is irrelevant.

## Part 2

1. The algorithm first calls `PL-BC-ENTAILS(KB, A)`. And since A is not known to be true in the knowlegde base, we need to check for clauses in the KB that can entail A.
2. In the `for loop` we find the clause $D \land B \land C \Rightarrow A$ exist in the KB.
3. Now we need to check for premises D, B and C with the function `CHECK-ALL(KB, c.PREMISE)`.\
    1. We first call `PL-BC-ENTAILS(KB, D)` and finds $(F \land E \Rightarrow D)$ in the KB. Since F and E are already known to be entailed, we can prove that the clause $(F \land E \Rightarrow D)$ is proven true. Therefore D is entailed by the KB.
    2. We then call `PL-BC-ENTAILS(KB, B)` and finds $(E \Rightarrow B)$, and since E is entailed, B is proven to be true and therfore is entailed by the KB.
    3. We then call `PL-BC-ENTAILS(KB, C)` and finds $(B \land E \land G \Rightarrow C)$. We have already proven that B and E to be entailed. Therefore we need to check if G is entailed by calling `PL-BC-ENTAILS(KB, G)`. We find that $(C \Rightarrow G)$, but this is a recursive problem since we need to prove C is entailed, which we orginally needed to prove. Therefore the algorithm will not terminate.

Since we found a problem with recursion, professor Smart's algorithm cannot answer if $KB \models$ A.

To fix this problem we can add a list as an extra paremeter to the functions `PL-BC-ENTAILS(KB, A, [])` and `CHECK-ALL(KB, premise, [])`. This list will keep track of symbols that are currently being checked in the for loop. This make sure if there is another way to prove C without G, it will find it, otherwise it will prove that KB does not entail A.

### The new extended pseudocode

```
function PL-BC-ENTAILS(KB, q, currentlyChecked) returns true or false
   inputs:
      KB, the knowledge base, a set of propositional definite clauses q, the query, a propositional symbol, and a list of symbols that currently are being checked

   if q is in currentlyChecked then return false
   if q is known to be true in KB then return true
   add q to currentlyChecked
      for each clause c in KB where q is in c.CONCLUSION do
         if CHECK-ALL(KB, c.PREMISE, currentlyChecked) then
            remove q from currentlyChecked
            return true
         else
            return false

function CHECK-ALL(KB, premise, currentlyChecked) returns true or false

   inputs: KB, the knowledge base, a set of propositional definite clauses premise, a set of propositional symbols, and a list of symbols that currently are being checked

   for each p in premise do
      if not PL-BC-ENTAILS(KB, p, currentlyChecked) then return false
      return true
```

With the addtion of having a list for currently checked symbols to the functions `PL-BC-ENTAILS` and `CHECK-ALL`. The algorithm will not be stuck with an infinite recursiv loop with C and G, since that it will check that C is already being checked when trying to prove the entailment of G. This will therefore fix the problem for professor Smart's algorithm to answer $KB \models$ A.
With the addtional changes to the algorithm we get the answer that `KB does not entail A`, since there exist a clause that is not proven to be entailed.