Mini-project 3
Spring Semester 2023

Logic and Discrete Mathematics

*R*oskilde University

Group members:
Vladyslav Horbatenko
Lucie Na... personal information

Roskilde University
April 2023

Contents

Mini-project 3 is a continuation of mini-projects 1 and 2. The formalities are unchanged. The deadline for handing in Mini-project 3 can be found at Moodle. Mini-project 3 consists of two tracks of questions, A and B, and you have to hand in exactly one of them

1 Question 1

Question 3B-1: Define the set of propositional formulas using a Type 2 (also called context-free) grammar, cf. page 789 in Rosen's book [1]. Hint: You might get inspiration by looking at the inductive definition of formulas at Example 10, page 301 in Rosen's book. Show that the language of your grammar contains the two formulas of questions 4 and 5 of Mini-project 2.

Answer for question 1:

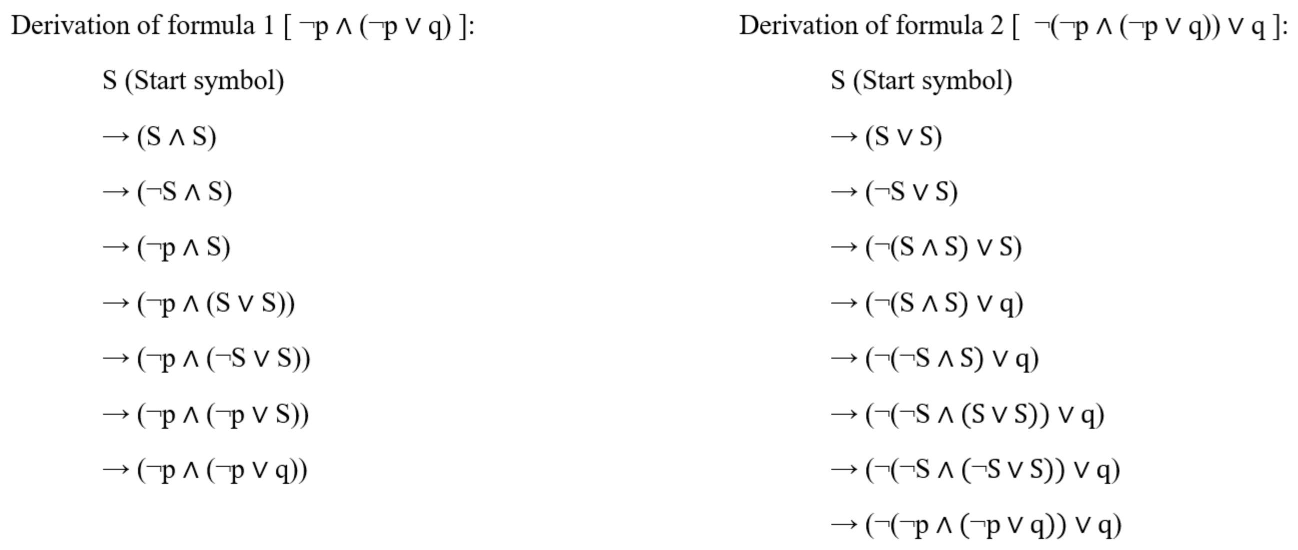
Question 4: $\neg p \land (p \implies q))$
Remove implication connective using $p \implies q \equiv \neg p \lor q$ to get: $\neg p \land (\neg p \lor q))$

Question 5: $(\neg p \land (p \implies q)) \implies q$
Remove implication connective using $p \implies q \equiv \neg p \lor q$ to get: $\neg(\neg p \land (\neg p \lor q)) \lor q$

We define a set of these formulas using Type 2 (context-free) grammar:

- G = {V, T, S, P}
- V = S ∪ T
- T = {p, q, ¬, (,), ∧, ∨}
- S = {S}
- P = {S → (S ∧ S), S → (S ∨ S), S → ¬ S}

By applying the defined production rules from the set P, we can see that both of the formulas are contained in the grammar we have defined.



Derivation of formula 1 [¬p ∧ (¬p ∨ q)]:

S (Start symbol)

→ (S ∧ S)

→ (¬S ∧ S)

→ (¬p ∧ S)

→ (¬p ∧ (S ∨ S))

→ (¬p ∧ (¬S ∨ S))

→ (¬p ∧ (¬p ∨ S))

→ (¬p ∧ (¬p ∨ q))

Derivation of formula 2 [¬(¬p ∧ (¬p ∨ q)) ∨ q]:

S (Start symbol)

→ (S ∨ S)

→ (¬S ∨ S)

→ (¬(S ∧ S) ∨ S)

→ (¬(S ∧ S) ∨ q)

→ (¬(¬S ∧ S) ∨ q)

→ (¬(¬S ∧ (S ∨ S)) ∨ q)

→ (¬(¬S ∧ (¬S ∨ S)) ∨ q)

→ (¬(¬p ∧ (¬p ∨ q)) ∨ q)

2 Question 2

Question 3B-2: Implement the informal decision procedure embodied in Question 3 of Mini-project 2 in the programming language Prolog (cf. Theorem 4, called soundness, and the remark immediately after Theorem 4 on what is called completeness). The Prolog program has to be included. In the interest of simplicity, you only need to consider propositional formulas built using the connectives $\neg$ and $\wedge$.

Answer for question 2:

```
% conjunction (and its negation)
solve([[and(P, Q)|Branch]|Rest], Updated) :-
    solve([[P,Q|Branch]|Rest], Updated).

solve([[not(and(P, Q))|Branch]|Rest], Updated) :-
    solve([[not(P)|Branch],[not(Q)|Branch]|Rest], Updated).

% disjunction (and its negation)
solve([[or(P, Q)|Branch]|Rest], Updated) :-
    solve([[P|Branch], [Q|Branch]|Rest], Updated).

solve([[not(or(P,Q))|Branch]|Rest], Updated) :-
    solve([[not(P),not(Q)|Branch]|Rest], Updated).

% negation
solve([[(not(not(X)))|Branch]|Rest], Updated) :-
    solve([[X|Branch]|Rest], Updated).

% when list of expressions is empty
solve([[ ] |Updated], [[]|Empty]) :-
    solve(Updated, Empty).

%if branch is empty
solve([ ],[ ]).

% program logic
solve([[X|Branch]|Updated], Final) :-
    solve([Branch], Empty),
    distribute(X, Empty, Overal),
    solve(Updated, Overal2),
    append(Overal, Overal2, Final).

% data distribution
distribute(X, [Branch|Rest], [[X|Branch]|Updated]) :-
    distribute(X, Rest, Updated).

distribute(_, [ ], []).
```

Listing 1: Prolog procedure

3 Question 3

Question 3B-3: Try out your Prolog program on the two formulas of questions 4 and 5 of Mini-project 2. Screen shots and brief explanations have to be included. If your answer to questions 4 and 5 in Mini-project 2 was not 100% correct, please redo them and include the answer in details.

Answer for question 3:

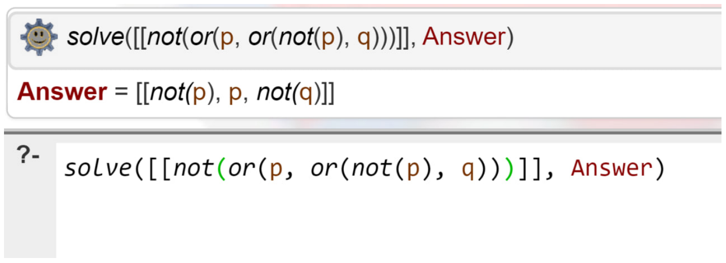
Let's take a look at formulas of questions from mini project 2.

In question 4 we used formula from exercise 1.2.11c: $\neg p \wedge (p \rightarrow q))$

	Formula	*Rules used*
We can rewrite the formula as:	$\neg\neg p \vee (\neg p \vee q)$	$p \rightarrow q \equiv \neg p \vee q$
and then as:	$p \vee (\neg p \vee q)$	$\neg(\neg p) \equiv p$

To prove that the formula in 1.2.11c is a tautology, we try to show that the negation of this formula $(\neg(p \vee (\neg p \vee q)))$ is not satisfiable. Now we will give this formula to our program. Our query will look like: *solve([[or(p, or(not(p), q))]], Answer)*

After we put this in our program we get this output:



```
solve([[not(or(p, or(not(p), q)))]], Answer)
Answer = [[not(p), p, not(q)]]
?- solve([[not(or(p, or(not(p), q)))]], Answer)
```

Figure 1: Mini project 2 Question 4 output

Output shows a contradiction p and $\neg p$ in the branch, meaning this branch is closed, so the formula is not satisfiable.

Answer for question 5:

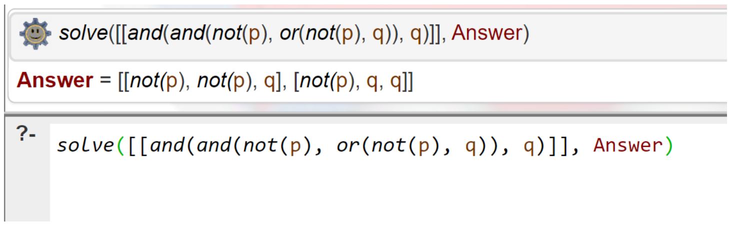
In question 5 we used formula from exercise 1.2.14: $(\neg p \wedge (p \rightarrow q)) \rightarrow \neg q$

To determine wheter the formula in 1.2.14 is a tautology, we try to show that the negation of this formula $(\neg((\neg p \wedge (p \rightarrow q)) \rightarrow \neg q))$ is not satisfiable. First, we remove indications from the formula in order to be able to work with it better:

	Formula	*Rules used*
We can rewrite the formula as:	$(\neg p \wedge (p \rightarrow q)) \wedge \neg\neg q$	$\neg(p \rightarrow q) \equiv p \wedge \neg q$
and then as:	$(\neg p \wedge (\neg p \vee q)) \wedge \neg\neg q$	$p \rightarrow q \equiv \neg p \vee q$
finally:	$(\neg p \wedge (\neg p \vee q)) \wedge q$	$\neg(\neg p) \equiv p$

Now, we try to determine whether the formula from 1.2.14 is a tauotology by proving that the negation of this formula is not satisfiable. Our query will look like:
solve([[and(and(not(p), or(not(p), q)), q)]], Answer)

After we put this in our program we get this output:



```
solve([[and(and(not(p), or(not(p), q)), q)]], Answer)
Answer = [[not(p), not(p), q], [not(p), q, q]]

?-  solve([[and(and(not(p), or(not(p), q)), q)]], Answer)
```

Figure 2: Mini project 2 Question 5 output

We ended with two branches that are open, which means that the formula in 1.2.14 is not a tauotology, since the negation of this formula is satisfiable.

4 Question 4

Question 3B-4: Try out your Prolog implementation on a formula which hasn't been considered in this course. The resulting tableau has to be described in Prolog notation as well as Priest's graphical notation. Screen shots and brief explanations have to be included.

Answer for question 4:

The formula we chose is $(p \land \neg q) \lor ((p \implies q) \land \neg p)$. For the sake of simplicity, we use the rule $p \implies q \equiv \neg p \lor q$ to obtain $(p \land \neg q) \lor ((\neg p \lor q) \land \neg p)$.

Solving this formula using set notation:

$$\begin{aligned}
(p \land \neg q) \lor ((\neg p \lor q) \land \neg p) \quad &\rightsquigarrow \quad \{\{p \land \neg q\}, \{(\neg p \lor q) \land \neg p\}\} \\
&\rightsquigarrow \quad \{\{p, \neg q\}, \{\neg p \lor q, \neg p\}\} \\
&\rightsquigarrow \quad \{\{p, \neg q\}, \{\neg p, \neg p\}, \{\neg p, q\}\}
\end{aligned}$$

Solving this formula using Priest's graphical notation:

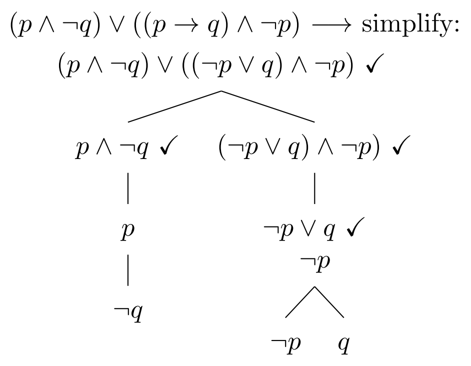
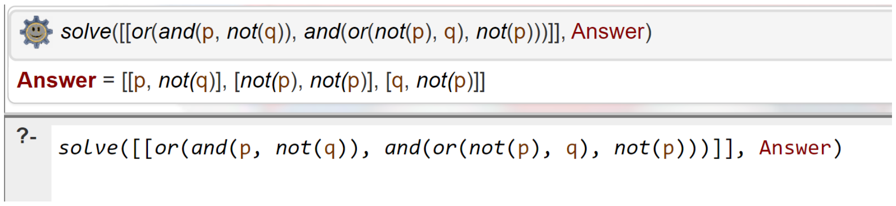

$(p \land \neg q) \lor ((p \to q) \land \neg p) \longrightarrow$ simplify:
$(p \land \neg q) \lor ((\neg p \lor q) \land \neg p)$ ✓
$p \land \neg q$ ✓
$(\neg p \lor q) \land \neg p)$ ✓
p
$\neg q$
$\neg p \lor q$ ✓
$\neg p$
$\neg p$
q

Figure 3: Priest's graphical notation

The tableau has 3 branches, with all 3 being open. This means that this formula is satisfiable.

Solving this formula using our Prolog Programm: As query we will use: *solve([[or(and(p, not(q)), and(or(not(p), q), not(p)))]], Answer)*



```
solve([[or(and(p, not(q)), and(or(not(p), q), not(p)))]], Answer)
Answer = [[p, not(q)], [not(p), not(p)], [q, not(p)]]
?- solve([[or(and(p, not(q)), and(or(not(p), q), not(p)))]], Answer)
```

Figure 4: Output for the formula using our Prolog program

Comparing the output of our Prolog program the results we got using the set notation Priest's graphical notation, we can see that they are the same.

References

[1] Kenneth H. Rosen. *Discrete mathematics and its applications: WITH MathZone*. McGraw Hill Higher Education, Maidenhead, England, 6 edition, 2006.