Assignment 1

02156 Logical Systems and Logic Programming, Fall 2023

September 24, 2023

Thomas Schiøler Hansen s214968

1 Table of contents

Contents

2 Problem 1

2.1 Problem 1.1

For the $\neg$, $\wedge$ and $\leftrightarrow$ operators I created the tables from the given semantics. In the truth tables I assume that X negated is also X, because we do not know what X is. For each of the truth tables I have created the corresponding classical truth table. Please see them below.

Table 1: Negation

A	$\neg A$
T	F
F	T
X	X

Comparing it to the classical logic of negation you can see that they are quite similar, just with the difference of the last row. Please see the classical logic for negation below:

Table 2: Classical logic - Negation

A	$\neg A$
T	F
F	T

Table 3: Conjunction

A	B	A $\wedge$ B
T	T	T
T	F	F
F	T	F
F	F	F
T	X	X
F	X	F
X	T	X
X	F	F
X	X	X

Comparing it to the classical logic for conjunction, one can see that they are quite similar in the sense that for every row except for the rows containing the X values. Please see the

classical truth table below:

Table 4: Classical logic - conjunction

A	B	A ∧ B
T	T	T
T	F	F
F	T	F
F	F	F

Table 5: Bi-implication

A	B	A ↔ B
T	T	T
T	F	F
F	T	F
F	F	T
T	X	X
F	X	X
X	T	X
X	F	X
X	X	T

Below please see that the classical truth table for bi-implication is again the exact same except for when there are values of X:

Table 6: Classical logic - Bi-implication

A	B	A ↔ B
T	T	T
T	F	F
F	T	F
F	F	T

When creating the truth table for A ∨ B, we can create a truth table of ¬(¬ A ∧ ¬ B), since they are equivalent:

Table 7: ¬(¬ A ∧ ¬ B))

A	B	¬ A	¬ B	¬(¬ A ∧ ¬ B)
T	T	F	F	T
T	F	F	T	T
F	T	T	F	T
F	F	T	T	F
T	X	F	X	T
F	X	T	X	X
X	T	X	F	T
X	F	X	T	X
X	X	X	X	X

Now that we have created the truth table we can create it for A ∨ B as seen below: X X -¿ X because it corresponds to A=B

Table 8: Disjunction

A	B	A ∨ B
T	T	T
T	F	T
F	T	T
F	F	F
T	X	T
F	X	X
X	T	T
X	F	X
X	X	X

If we compare it to the classical logic for disjunction as seen below, one can see that they are the same up until X gets involved. The new logic just has some extra rows due to the X value.

Table 9: Classical logic - disjunction

A	B	A ∨ B
T	T	T
T	F	T
F	T	T
F	F	F

We follow the same process now when creating the truth table for A $\rightarrow$ B. We know it is equivalent to A $\leftrightarrow$ (A $\wedge$ B), so we start by making the truth table for A $\leftrightarrow$ (A $\wedge$ B):

Table 10: A $\leftrightarrow$ (A $\wedge$ B)

A	B	A $\wedge$ B	A $\leftrightarrow$ (A $\wedge$ B)
T	T	T	T
T	F	F	F
F	T	F	T
F	F	F	T
T	X	X	X
F	X	F	T
X	T	X	T
X	F	F	X
X	X	X	T

Using this table we can construct the truth table for A $\rightarrow$ B:

Table 11: Implication

A	B	A $\rightarrow$ B
T	T	T
T	F	F
F	T	T
F	F	T
T	X	X
F	X	T
X	T	T
X	F	X
X	X	T

Again, the same goes for implication, it is the same in the classical sense except for the rows with X values:

Table 12: Classical logic - Implication

A	B	A $\rightarrow$ B
T	T	T
T	F	F
F	T	T
F	F	T

2.2 Problem 1.2

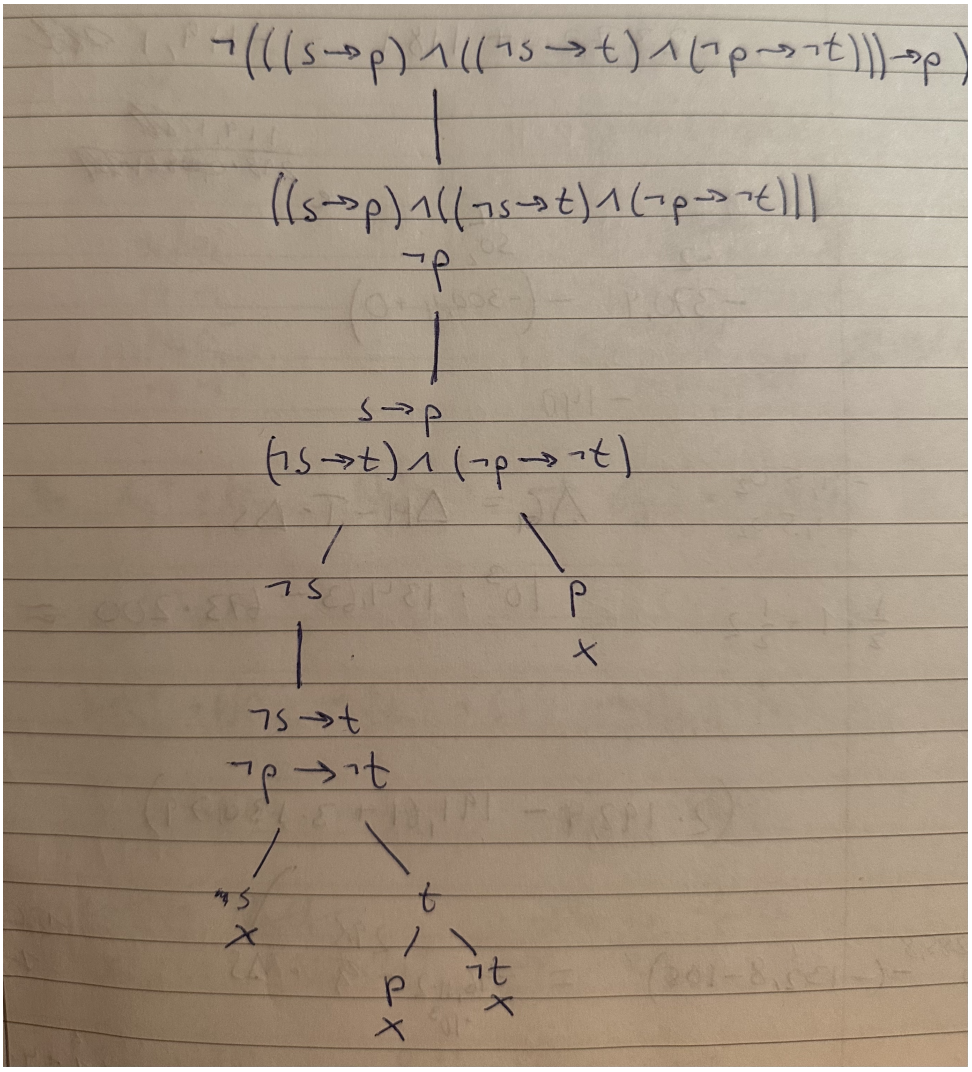
I have constructed the truth table below for the truth values T, F and X

P	$\neg$ P	$\neg$ P $\wedge$ P
T	F	F
F	T	F
X	X	X

3 Problem 2

3.1 Problem 2.1

Below please see the picture of the tableau that I have created. All the branches close for the negated formula, meaning that the original formula is valid:



3.2 Problem 2.2

We have been given the following formula, I will call it F1:
$((s \rightarrow p) \wedge ((\neg s \rightarrow t) \wedge (\neg p \rightarrow \neg t))) \rightarrow p$

Through the logical equivalence below be want to show that the formula is equivalent to the formula that is shown under the equivalence, I will call that formula F2.
$A \rightarrow B \equiv \neg B \rightarrow \neg A$

$((\neg p \rightarrow \neg s) \wedge ((\neg s \rightarrow t) \wedge (t \rightarrow p))) \rightarrow p$
In order to show the equivalence, I need to make the two formulas the same. I will do that using the logical equivalence given by doing two operations:
First, I will change $\neg p \rightarrow \neg s$ in F2 to $s \rightarrow p$ as we know that they are equivalent.
$((s \rightarrow p) \wedge ((\neg s \rightarrow t) \wedge (t \rightarrow p))) \rightarrow p$
Now I will change $t \rightarrow p$ to $\neg p \rightarrow \neg t$ as they are equivalent.
$((s \rightarrow p) \wedge ((\neg s \rightarrow t) \wedge (\neg p \rightarrow \neg t))) \rightarrow p$
Now F2 is the same formula as F1 and we can, therefore, say that they are equivalent because we could make them the same using the logical equivalence.

4 Problem 3

4.1 Problem 3.1

I have successfully created the predicate "member2" without using any other predicates - please see the attached file "assignment1.pl".
Multiple tests have been issued in order to determine if my solution is efficient. When the consecutive elements are in the start of the list. Below please see three tests that test the normal cases:
?- member2(a, [a, a, b, a, c]).
true ;
false.
It is expected to return true as 'a' appears twice in the start which is covered by a base case in the code. Below is a test where the consecutive elements appear anywhere in the middle of the list:
?- member2(a, [b, a, c, a]).
false.
Here is returns false as it was expected to because a does not appear consecutively in the list.

Lastly, we will test for if a appears consecutively anywhere in the middle of the list:
?- member2(a,[b, c, a, a, d]).
true ;
false.
It also returns true, as it is expected to.
Now we will test some special cases: ?- ?- member2(a, []).
false.
Here we test with an empty list and it returns false as it is expected to do as there are no consecutive elements in an empty list. The last special case test is when there is only one element in the list:
?- member2(a, [a]).
false.
This test returns false as it should, even though there is only one element in the list as well as the element being the same as the one we search for.
We will now create two tests using variables:
?- member2(X, [X, X, Y, Z]), X = a, Y = b, Z = c.
X = a,
Y = b,
Z = c ;
false.
As expected this test succeeds with X = a, Y = b and Z = c. The last test:
?- member2(X, [Y, Z, a]), X = a, Y = b, Z = c.
false.
This test fails because 'a' does not appear consecutively in the list. This is as expected.

4.2 Problem 3.2

Please refer the attached file "assignment1.pl", to see my solution of expressing the predicate member2 only by using append. Again, in order to test the code I have issued a the same tests as before to see if the code runs properly. Firstly, the normal cases:
?- member2a(a, [a, a, b, c]).
true ;
false.
This returns true as is expected because a appears consecutively in the start of the list. Please see the next test:
?- member2a(a, [b, a, c, a]).
false.

This test fails as it should because a is not in the list consecutively. We will now test for when a appears consecutively in the middle of the list:

member2(a,[b, c, a, a, d]).

true ;

false.

It returns true, which is what would be expected. We will now test for some special cases:

?- member2a(a, []).

false.

It fails when the list is empty, which is the same as before and is what we expected. Now for the next test where a is the only element in the list:

member2a(a, [a]).

false.

It fails as a is not in the list consecutively, which again is what we expected. Lastly, we will now test with variables:

?- member2a(X, [X, X, Y, Z]), X = a, Y = b, Z = c.

X = a,

Y = b,

Z = c ;

false.

Again, the output is as expected, that it succeeds, because X appears twice in the list. Lastly:

?- member2a(X, [Y, Z, a]), X = a, Y = b, Z = c.

false.

It returns false as it should since a does not appear consecutively.