DTU Course 02156 Logical Systems and Logic Programming

Mandatory Assignment 1 — Deadline Sunday 24/9 23:59

MUST BE SOLVED INDIVIDUALLY

You are only allowed to get help from the teacher and the teaching assistants.

You may use the notes and definitions on the first page of the sample exams.
You are allowed to use your computer and there is no 2 hours time limit.

For the whole assignment you must submit exactly 2 files on DTU Learn:

1. A single PDF file with extension `.pdf` with the report.
2. A single Prolog file with extension `.pl` with the programs.

Absolutely no ZIP files, no text processing documents or any other file formats.

The report should not contain any program listings (just refer to the Prolog file).

The programs should load in SWI-Prolog without any errors or warnings.
The programs must be properly documented with comments.
If possible use only the ISO Prolog features of SWI-Prolog covered in the course.

All programs must be tested and the tests must be included in the report.

In particular:

- Test a few normal cases.
- Test the special cases.
- Test with variables only (if the instantiation pattern allows variables).

Show the Prolog queries and the corresponding answers — and keep explanations short.

Problem 1 (50%)

The aim is to investigate a many-valued logic that can have any number of truth values as opposed to classical logic with the two truth values **T** and **F**.

We use $n \geq 2$ to denote the number of truth values.

We use $[[A]]$ to denote the truth value of the formula A.

We define the semantics of our logic as follows for negation, conjunction and bi-implication:

$$[[\neg A]] = \begin{cases} \mathbf{T} & \text{if } [[A]] = \mathbf{F} \\ \mathbf{F} & \text{if } [[A]] = \mathbf{T} \\ [[A]] & \text{otherwise} \end{cases}$$

$$[[A \wedge B]] = \begin{cases} [[A]] & \text{if } [[A]] = [[B]] \\ [[B]] & \text{if } [[A]] = \mathbf{T} \\ [[A]] & \text{if } [[B]] = \mathbf{T} \\ \mathbf{F} & \text{otherwise} \end{cases}$$

$$[[A \leftrightarrow B]] = \begin{cases} \mathbf{T} & \text{if } [[A]] = [[B]] \\ [[B]] & \text{if } [[A]] = \mathbf{T} \\ [[A]] & \text{if } [[B]] = \mathbf{T} \\ [[\neg B]] & \text{if } [[A]] = \mathbf{F} \\ [[\neg A]] & \text{if } [[B]] = \mathbf{F} \\ \mathbf{F} & \text{otherwise} \end{cases}$$

Additionally we define disjunction and implication using the following equivalences:

$$A \vee B \;\equiv\; \neg(\neg A \wedge \neg B) \qquad\qquad A \rightarrow B \;\equiv\; A \leftrightarrow (A \wedge B)$$

We can use truth tables as in classical logic (although not always the same truth tables).

Let $n = 3$ in the following questions.

Question 1.1

Compare the logic to classical logic by creating truth tables for the operators $\neg$, $\wedge$, $\vee$, $\rightarrow$ and $\leftrightarrow$ using the truth values **T**, **F** and **X**.

Question 1.2

Calculate the truth values of the formula $\neg p \wedge p$ when p has the truth values **T**, **F** and **X**.

A historical comment:

The many-valued logic — a so-called paraconsistent logic — was first considered at the Scandinavian Conference on Artificial Intelligence in 2001 by Jørgen Villadsen.

More information is available here: `http://people.compute.dtu.dk/jovi/poster/`

Problem 2 (25%)

Consider the following formula:

$$((s \to p) \land ((\neg s \to t) \land (\neg p \to \neg t))) \to p$$

Question 2.1

Use refutation and the systematic construction of a semantic tableau.

State whether this shows that the formula is valid or not.

Question 2.2

Show that the formula is equivalent to $((\neg p \to \neg s) \land ((\neg s \to t) \land (t \to p))) \to p$ using the following logical equivalence only:

$$A \to B \equiv \neg B \to \neg A$$

A historical comment:

The formula corresponds to the following argument in natural language:

If Joe studies then he passes the exam.
If Joe does not study then he has a good time.
If Joe does not pass the exam then he does not have a good time.
Therefore Joe passes the exam.

A similar argument — albeit with a college student called Alfred — is discussed on the first page of *Logic: Techniques of Formal Reasoning (Second Edition)* by Donald Kalish, Richard Montague and Gary Mar (Oxford University Press 1980).

Problem 3 (25%)

Consider a program `member2(?Elem,?List)` that succeeds if and only if `Elem` appears as two consecutive members in `List` (multiple answers should be possible on backtracking).

Question 3.1

Express the predicate `member2` not using any other predicate (recursion allowed).

Question 3.2

Express the predicate `member2` using the predicate `append` only (no recursion allowed). It is pretty simple. Call the defined predicate `member2a` (`member2` via `append`).