02157 Functional programming

Michael R. Hansen
DTU Informatics

November 10, 2011
Revised November 6, 2013

Mandatory assignment: Solver for Propositional Logic

This is the second mandatory assignment where you should make a simple solver for Propositional Logic. The purpose of the exercise is to cover basic program construction principles involving finite trees and sets. The requirements for your submission are as follows:

- Your solution should be handed in **no later than Tuesday, November 29, 2011**. The mandatory assignments can be solved individually or in groups of 2 or 3 students. If the students in a group do not contribute equally, the role of each student must be explicitly stated in the report.
- **One** member of each group must hand in the assignment using the "Assignments" link in the course group on CampusNet. Two individual files must be uploaded:
  - A PDF-file 'assignment2.pdf' containing just **one** page with the names and study numbers for all members of the group. This page must also contain:
    - A formulation of the Knight-Knave puzzle in propositional logic (Question 7) together with a solution.
  - A file 'Prop.fs' containing your program solution to this assignment together with suitable tests.

Propositional Logic

In this assignment you shall consider formulas of propositional logic, also called *propositions*, which are generated from a set of *atoms* $P, Q, R, \ldots$ by the use of the well-known operators: *negation* $\neg$, *disjunction* $\vee$ and *conjunction* $\wedge$. An atom can be either true or false, and the meaning of each the connectives is:

- $\neg A$ is true if and only is A is false,
- $A \vee B$ is true if and only if A is true or B is true or both are true, and
- $A \wedge B$ is true if and only if both A and B are true,

where A and B can be arbitrary propositions.

Any other propositional operator can be expressed using $\neg, \vee$, and $\wedge$. (Actually negation together with just disjunction or just conjunction would suffice.) Implication and biimplication (or equivalence), for example, are definable as follows

$$\begin{array}{lll} A \Rightarrow B & \text{is expressed as} & (\neg A) \vee B \\ A \Leftrightarrow B & \text{is expressed as} & (A \Rightarrow B) \wedge (B \Rightarrow A) \end{array}$$

Questions 1

You should define a type `Prop` for propositions so that the following are values of type `Prop`:

- `A "p"` represents the atom p,
- `Dis(A "p", A "q")` represents the proposition $p \vee q$.
- `Con(A "p", A "q")` represents the proposition $p \wedge q$.
- `Neg(A "p")` represents the proposition $\neg p$.

Question 2

A proposition is in *negation normal form* if the negation operator just appears as applied directly to atoms. Write an F# function transforming a proposition into an equivalent proposition in negation normal form, using the de Morgan laws:

$$\begin{array}{lcl} \neg(A \wedge B) & \Leftrightarrow & (\neg A) \vee (\neg B) \\ \neg(A \vee B) & \Leftrightarrow & (\neg A) \wedge (\neg B) \end{array}$$

and the law: $\neg(\neg A) \;\Leftrightarrow\; A$.

Question 3

A *literal* is an atom or the negation of an atom and a *basic conjunct* is a conjunction of literals. A proposition is in *disjunctive normal form* if it is a disjunction of basic conjuncts. Write an F# function which transforms a proposition in negation normal form into an equivalent proposition in disjunctive normal form using the laws:

$$\begin{array}{lcl} A \wedge (B \vee C) & \Leftrightarrow & (A \wedge B) \vee (A \wedge C) \\ (A \vee B) \wedge C & \Leftrightarrow & (A \wedge C) \vee (B \wedge C) \end{array}$$

Question 4

We shall use a set-based representation of formulas in disjunctive normal form. Since conjunction is commutative, associative and satisfies that $(A \wedge A) \Leftrightarrow A$ it is convenient to represent a basic conjunct bc by its set of literals `litOf`(bc).

Similarly, we represent a disjunctive normal form formula A:

$$bc_1 \vee \ldots \vee bc_n$$

by the set

$$\texttt{dnfToSet}(A) = \{\texttt{litOf}(bc_1), \ldots, \texttt{litOf}(bc_n)\}$$

that we will call the *dns set* of A.

Write F# declarations for the functions `litOf` and `dnfToSet`.

Question 5

A set of literals *ls* (and the corresponding basic conjunction) is *consistent*, if for no atom p we have have that both p and $\neg p$ are literals in *ls*. Otherwise, *ls* (and the corresponding basic conjunctions) is *inconsistent*. An inconsistent basic conjunct is false regardless of the truth values assigned to atoms. Removing it from a disjunctive normal form formula will therefore not change the meaning of that formula.

Declare an F# function `isConsistent` that checks the consistency of a set of literals. Declare an F# function `removeInconsistent` that removes inconsistent literal sets from a dns set.

Question 6

A proposition is *satisfiable* if it is true for some assignment of truth values to the atoms.

A formula in disjunctive normal is satisfiable when one (or more) of its basic conjunctions are. Therefore, the satisfying assignments of a proposition can be inspected from the consistent literal sets of its disjunctive normal form.

Declare an F# function `toDNFsets` that transforms an arbitrary proposition into a dns set with just consistent literal sets.

Declare a function `impl` A B for implication $A \Rightarrow B$ and a function `iff` A B for biimplication $A \Leftrightarrow B$.

Use `toDNFsets` to determine the satisfying assignments for the following two formulas:

- $((\neg p) \Rightarrow (\neg q)) \Rightarrow (p \Rightarrow q)$
- $((\neg p) \Rightarrow (\neg q)) \Rightarrow (q \Rightarrow p)$

Question 7

In this question you shall solve a *Knights and Knaves* puzzle, which is a kind of puzzle originating from the logician R. Smullyan. The general theme addresses an island that is inhibited by two kinds of citizens: Knights, who always tell the truth, and knaves, who always tell lies. On the basis of utterances from some inhabitants you must decide what kind they are.

You may find many Knight and Knave puzzles on the internet. The following is originates from `http://www.homeschoolmath.net/reviews/eimacs-logic.php`.

Three inhabitants, A, B and C, are talking.

- A says: "All of us are knaves."
- B says: "Exactly one of us is a knight."

To solve the puzzle you should determine: What kinds of citizens are A, B and C?

Solve this puzzle by modelling the two utterances above in propositional logic and using `toDNSsets` to find the satisfying assignments. (Hint: Use A to describe that A is a knight and $\neg A$ to describe that A is a Knave.)

Question 8

The satisfiability problem for propositional logic is an NP-complete problem, and the above transformation to disjunctive normal form proposition or to a dns set has in the worst case an exponential running time.

The disjunctive normal form of the following proposition:

$$(P_1 \vee Q_1) \wedge (P_2 \vee Q_2) \wedge \cdots \wedge (P_n \vee Q_n) \tag{1}$$

will, for example, have 2^n basic conjuncts.

Declare an F# function `badProp` n that computes the F# representation of (1).

Compute `toDNFsets(badProp` n`)` for a small number of cases and check that the resulting sets indeed have 2^n elements.