Compulsory Assignment: Propositions

The point of this exercise is to construct a F# module providing a type for a proposition and related functions.

A proposition is a boolean formula where you can use constants, Boolean operators, and variables. There are only two constants: true and false. As boolean operators we will in this exercise limit to not, and, or, equal, implies, and exclusive or with the usual semantic. Variables are simple names. You must first define a type for a proposition such that it is possible to declare these in F#. For instance; the proposition

```
a or ((b and not a) => c)
```

Could be declare in F# as

```
let p = Or(Variable "a",
           Implies(And(Variable "b", Not (Variable "a"),
                   Variable "c")
          )
```

The specific naming of operators are not mandatory.

You must construct several functions related to the proposition. These are:

Evaluate
The function will compute the value of a proposition in given state or environment. The environment is a binding of values (either true or false) to all variables used in the proposition. You may define your own type for environment or just use Map<string, Boolean>. The signature for the Evaluate function will be:

```
Evaluate: (p: Proposition) -> (env: Map<string, bool>) -> bool
```

Verify
The function can be used to check that all variables used in proposition are defined in an environment. The signature is

```
Verify: (p: Proposition) -> (env: Map<string, bool>) -> bool
```

Tautology
A tautology is a proposition that is true in all possible environments. For instance, "a or not a" is a tautology. You must implement a function that will return true if and only if a proposition is a tautology. The signature for the function is:

```
IsTautology: (p: Proposition) -> bool
```

Satisfiable
You must declare a function that can check if there exists an environment where a proposition will be true. The signature for the function is:

```
IsSatisfiable: (p: Proposition) -> bool
```

Equality
You must declare a function that can check if two propositions are equal in the sense that that they will evaluate to the same value for all environments. The signature for the function is:

```
IsEqual: (p: Proposition) (q: Proposition) -> bool
```

ToString
You must declare a function that convert a F# proposition to a string.

Names used
You must declare a function that will return a list of strings; that is all the names of the variables used in a proposition. The signature for the function is:

```
NamesUsed: (p: Proposition) -> string list
```

You must hand-in a report containing all the source code well documented not later than Monday 29/11 at 9am either individually or in groups of up to three students. The report must contain

- A front-page with title, and full names on all students.
- Page number on all pages.
- A section on state-of-delivery.
- A section for each module in your solution. All source code must be in the pdf document and it is important that it is documented properly.
- A section on testing. Describe how you have tested your module and provide snippets of test code in the report.

Please hand-in your solution by email to oe@easv.dk with the solution in a .pdf document attached and your solution in a .fs file (or zip if several files) (or provide a url to a Github repository containing your solution). Upon return of your solution you will have feedback and a grade, which is either approved or failed. In absence of approval, you will have a second chance to get it approved.

Happy coding!

Ole Eriksen – oe@easv.dk