Computational Logic 2018-2: Lab report 1

Luis Daniel Aragon Bermudez 416041271

February 8th, 2017

Contents

Propositional calculus

This haskell module implements some basic functions of the propositional calculus. It also provides the data type `Prop` that implements a grammar for propositional logic.

This module also defines the interpretation (`interp`) function that recursively assigns a boolean value to a proposition given a state mapping. This is achieved by implementing the abstract type `type State` which we define as a list of strings. This list should itself be considered a partial mapping of propositional variables to truth states. If a string `s` is a member of a State, then we say that the state of a propositional variable `Var s` inside a WFF (data `Prop`) is true, else we say it's false or equivalently we say that `Neg (Var s)` is true.

We also define a `model` function implemented as an eta reduction idiom of the `interp` function. It emulates the notion of models within propositional calculus, which is to say that if φ is a WFF, a state I is a model of φ if and only if $I(\varphi) = \top$.

Other functions that are implemented in this module are:

- tautology. Determines if a proposition is a tautology, i.e. every interpretation of said proposition returns true.
- vars. Returns the set (as a list) of variables within a proposition.
- equivProp. Determines if two propositions are equivalent. By the rules of propositional logic: $\varphi \equiv \psi \iff \models \varphi \leftrightarrow \psi$
- logicConsequence. Determines if the proposition provided as the second argument is a logical consequence of the premises (list of propositions) provided as the first argument. It does this by applying the refutation principle: $\Gamma \models \varphi \iff \Gamma \cup \neg\{\varphi\}$ is unsatisfiable.

Loading the library

If you have stack, use `stack ghci src\pract1_lc20182.hs` on Windows or `stack ghci ./src/pract1_lc20182.hs` on mac or Linux in order to load the functions on the interactive shell. If you have ghci installed though, you can skip the use of `stack` in the previous commands. In Linux, for example, the script can be loaded by typing the following command:

```
$ ghci src/pract1_lc20182.hs
```