

# Proposition Generator

Pierre Misse (Hogo), Théo Rogliano (Alisu)

December 2018

## Abstract

CLisp program is used to transform prefixed first order logical proposition to their declaration for the Smalltalk program in the `../pharo/` directory.

## 1 Introduction

The Clisp program is necessary to run the script. You may have to `chmod 111 propGenerator.lisp` to be able to execute it.

A simple exemple would be:

$(Or (P x) (P q))$  which becomes:

```
Or new :
  (Predicate new: 'P' fromList :
    (LinkedList new
      add: (Term new: 'x ');
      yourself)
  )
rightProp :
  (Predicate new: 'P' fromList :
    (LinkedList new
      add: (Term new: 'q ');
      yourself)
  )
```

It allows a more general syntax, and to generate directly into the target language implementation the object initialization, which is tedious to write by hand.

## 2 Usage

Put in the input file ("input.prop" by default) the proposition(s) that you wish to transform. Then just run the script `./propGeneration`. The output will be displayed on the standard output, to be able to pipe it to another program (such as a clipboard one).

## 3 How to write a proposition

The proposition are wrote in a prefixed lisp way. Everything is case INsensitive.

### 3.1 Predicates the predicates and terms

A predicate is also surrounded by parenthesis.

$$(P x)$$

The terms in a predicate can either be a simple term such as in the previous example or a Function term.

$$(P (f x))$$

A predicate can also be empty, but still needs the parenthesis wrapping it up

$$(P)$$

The terms of a Predicate/Function term are both variadic

$$(P (f x y z) z (g s r))$$

Note that in a proposition, the entities which are under an Operator or a Quantifier will be considered as Predicate, and the ones inside a Predicate will be considered as function terms (We can have this property since we're in first order logic and not in higher-order logic).

### 3.2 Operators

It's usually an operator followed by one or two predicates. (And Predicate Predicate) (Not Predicate)

The operators aren't variadic.

### 3.3 Quantifier

In the same spirit:

$$(Exist terms Prop)$$

The terms in the quantifiers are variadic, followed by a unique proposition.

$$(Exists x y z Prop)$$

The terms can only be simple ones, no function will be recognize here. So  $(Exists (f x) Prop)$  DOESN'T work.

## 4 Configuration

The following things are configurable in the "configuration.lisp" file:

- The list of operator's symbols is modifiable, and variadic. /! The symbols are case INsensitive.
- The list of the operator's classes is modifiable without having to look in this tool's code.

Those lists aren't implemented yet for the constants.

- By default, the generated object is generated on several lines, with a correct indentation. A compact mode is also available ("compact-mode" in the indent section).
- The appearance of the indent is modifiable through the "indent-string" property, in the indent section.