COMP325 - Interpreter 2 Redux

Fall 2015

Due Friday 10/30 at the start of class

There were lots of problems in the second interpreter and understanding that material, both conceptually and in terms of Pyret-based language implementation, is vital to your long term success in this course. So, you have the chance to recoup some lost points on the interpreter and desugarer. You'll be given a reduced set of expressions definitions for which you must write the appropriate desugarer and interperter. This assignment optional. If you're satisfied with your score, then you can skip this and keep that score. The interpreter and desugarer described below are each worth a maximum of 5 points each. The points will be applied to your existing interpreter and desugarer scores. This assignment cannot raise those scores above 20 points. If you're current score is at or above 15, then you can raise it 20. If it's below 15, then the max you can achieve through this assignment is 5 more than your current score.

1 Interpreter

Write and fully test the an environment-based interpreter for the *ExprC* type given. Your interpreter must meet the following requirements:

1. Catch the run-time error that occurs when a function is applied to too many or too few arguments.
2. Have complete test coverage, both in terms of code and logical cases, including testing for run-time errors.
3. Use the Pyret high-order functions like *map* and *fold* whenever and where ever possible. No DIY recursion on lists.

You'll need to start the function from scratch and fully define and implement any necessary helpers and auxiliary data definitions.

2 Desugarer

You must write two desugar functions: *desugar-expr* and *desugar-def*. The former desugars and ExprExt to ExprC and the later a FunDefExt to FunDefC. Your desugaring code must meet the following requirements:

1. The *andExt* form should ultimately reduce to an *ifC* that properly carries out short-circuiting semantics for the and operator.
2. The multi-branching *condExt* should reduced to nested *ifC* expressions. Pay attention to the preconditions on the structure of the conditional's list of clauses.
3. Have complete test coverage, both in terms of code and logical cases, for the functions.
4. Use the Pyret high-order functions like *map* and *fold* whenever and where ever possible. No DIY recursion on lists.

You'll need to start the functions from scratch and fully define and implement any necessary helpers and auxiliary data definitions.