EGO'S INTERPRETER
User's Manual

Andi Bejleri (bejleri@cli.di.unipi.it)

December 19, 2005

Introduction

This document introduces the implementation of EGO'S INTERPRETER. Other documents give details about the underlying language Ego; see section References.

Getting Started

The distribution is available for download as http://www.cs.cmu.edu/∼ aldrich/ego/ego-0_1.tar.gz.

EGO'S INTERPRETER is written in Java and requires a Java runtime which supports Java 1.1.1 or higher. The JavaSoft web site describes how to download a Java runtime. You must be able to run Java cleanly from the command line (as the program java). That may involve setting the PATH, CLASSPATH, and JAVA_HOME environment variables. See the Java documentation for more details.

Unzip and untar the EGO'S INTERPRETER distribution ("%" is your command prompt):

```
% gunzip ego-0_1.tar.gz
% tar xf ego-0_1.tar
```

This produces a directory called ego-0_1 which contains all the files you'll need to install and run the interpreter. Now you are ready to install ego-0_1/install. Run the script ego-0_1/ego to launch EGO'S INTERPRETER; a text mode interface will appear to type the program and then press <ENTER> to execute it. This is EGO'S INTERPRETER, version 0.1. If you have difficulty or questions about installing or running EGO'S INTERPRETER, please contact the author and I will try to assist you.

Interaction with the Interpreter

This section demonstrates many of the features of the interpreter by way of a sample session. Text typed by the user appears in bold after the "-" prompt. Additional commentary appears on the right.

Part 1: Simple values	
At the actual implementation there are not considered simple values	
Part 2: Variables	
-x	Variables are not used as local ones but just for abstract parametrization.
-apply(fn x : |*Object.* <> |**.x, clone(Object))**	Assign an object to x. Like ML this definition hides any previous bindings to the identifier x, but does not destroy them. Earlier uses of x still refer to the previous binding.
Part 3: Method Invocations	
-WebPhoneBook.prepareNew	A simple example of a method invocation. *WebPhoneBook.prepareNew* is a built-in function which is returned.
-apply(WebPhoneBook.prepareNew, clone(Entry).addMethod(name, fn self:|entry|. andi).addMethod(phone, fn self:|entry|. 56183))	Adds a new entry to the WebPhoneBook.
Part 4: Objects and Parents	
-clone(Object)	The object keyword is an expression that constructs a fresh object in the system. Think of it as the expression new Object() in Java. Every object has a printed version by the toString method.
-clone(x)	x with parent y, clones x and adds an inheritance link from this clone to y. The value of the expression is the cloned object.
-x.delegate(y)	Inheritance links can be established between any linear object, who inherit,and nonlinear object, parent.
-self.addMethod(ok, *fn self: |default|.curEntry)	Adds a new method to the receiver. In this example it adds to **self** (**this** in Java) the ok method. The receiver has a linear type.
-change_linearity(obj)	Changes the linearity from linear to nonlinear to **obj**. The modified object cannot change its interface anymore.
Part 5: Methods	
fn x : |*Object.* <> |**.*fn y :** |*DEL*|**.x.delegate(y)**	This method declaration demonstrates several important features of the language: - Methods can have any number of arguments. - Methods can have linear or nonlinear types.

Abstract Syntax

This section presents an abstract syntax for the Ego language. Words in capital represent tokens of the language.

p ::= e

e ::= VAR
| FN VAR COLON LINE tau LINE DOT e
| BANG FN VAR COLON LINE tau LINE DOT e

| e DOT VAR
| e DOT DELEGATE LPAREN e RPAREN
| CLONE LPAREN e RPAREN
| e DOT ADDMETHOD LPAREN m RPAREN
| APPLY LPAREN e COMMA e RPAREN
| CHANGE UNDERSCORE LINEARITY LPAREN e RPAREN

m ::= VAR COMMA e

tau ::= VAR
| VAR DOT LANGLE r RANGLE
| VAR DOT BANG LANGLE r RANGLE
| tau LINEAR tau
| tau ARRAY tau

r ::=
| b SEMICOLON SUPER COLON tau

b ::= VAR COLON tau COMMA b
|

Finding More Information

How to Contact

If you discover a bug in the implementation of Ego's Interpreter, please contact me, Andi Bejleri.

Implementation details

Ego's Interpreter consists of approximately 1300 lines of Java. It relies on JavaCUP for lexical analysis and parsing (but users need not install those packages in order to run Ego's Interpreter).

References

Ego: Controlling the Power of Simplicity
by Andi Bejleri, Jonathan Aldrich and Kevin Bierhoff

A Type-checked Prototype-based Model with Linearity
by Andi Bejleri. Published as Carnegie Mellon Technical Report CMU-ISR-04-142, December 2004.