A principled approach to REPLs*

*Read-eval-print-loops, consoles, interactive shells, notebooks, command-lines



Joint work with Pierre Jeanjean, Benoit Combemale, Olivier Barais (INRIA, University of Rennes)

Problem: REPLs are popular tools, but their semantics is not well-defined

Solution: see REPLs as a (modular) *language extension*

How: define a *sequential* operator ";" and *compose* semantics.

REPLs: old but popular Very diverse design space:



Sequential languages:

"A language is sequential if the concatenation of two programs is again a program"





Sequentialize: $p_1 \stackrel{o}{_9} p_2$ syntax Cmd = Op | Print | assoc Cmd Cmd ; Conf eval((Cmd)`<Cmd c1> <Cmd c2>`, Conf c) = eval(c2, eval(c1, c)); 0 rascal-mpl.org

MiniJava Notebook

QL: questionnaire DSL

eFLINT: norms DSL



• src — java • -bash — 51×31 estion : prepend guestion	
ddress> (Question) : replace guestion	
pr> : evaluate expression	LII for to
> = <value> : update a question</value>	UTION
eta commands it : exit the REPL lp : show help v : show current environment ml : show web rendering rm : print the form nder : render the UI ace : show eval history load tax { id you sell a house in 2010? [false] id you buy a house in 2010? [false] id you enter a loan? [false] html ving visual content at [http://localhost:9050] asSoldHouse = true { id you sell a house in 2010? [true] id you sell a house in 2010? [frue] id you sell a house in 2010? [false] hat was the selling price? [0] rivate debts for the sold house: [0] alue residue: 0	 Did you sell Did you buy Did you enter What was the self Value residue Value residue Value residue (0) Did you (0) Did you (1) Did you boolean (2) Did you (3) if (hat sell) (3.1) house (3.2) (sell)
U	

Ċ Å 🗗 + localhost

xOfficeExample

a house in 2010? / a house in 2010? ter a loan? he selling price? ts for the sold house:

xOfficeExample

- u sell a house in 2010? hasSoldHouse:
- u buy a house in 2010? hasBoughtHouse:
- u enter a loan? hasMaintLoan: boolean
 - asSoldHouse)) What was the selling price? lingPrice: integer
 - Private debts for the sold
 - e:privateDebt: integer] Value residue: valueResidue: integer =
 - lingPrice privateDebt)
- #0 > Fact person. Placeholder parent, child For person new fact-type person no enabled actions or events #3 > +person(Alice). +person(Bob) // introduce persons +"Alice":person +"Bob":person no enabled actions or events #5 > Fact parent-of Identified by parent * child new fact-type parent-of no enabled actions or events #6 > +parent-of(Alice,Bob) +("Alice":person, "Bob":person):parent-of no enabled actions or events #7 > Act call-for-help Actor child Recipient parent **Holds when** parent-of() new fact-type call-for-help +("Bob":person, "Alice":person):call-for-help
- enabled actions & events:
- 1. ("Bob":person, "Alice":person):call-for-help

van Binsbergen, L.T, Verano Merino, M, Jeanjean, P, van der Storm, T, Combemale, B, & Barais, O. (2020). A principled approach to REPL interpreters. In Onward! 2020 - Proceedings of the 2020 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software, Co-located with SPLASH 2020 (pp. 84–100). doi:10.1145/3426428.3426917