TESTING STATEFUL SYSTEMS WITH QUICKCHECK AND PROLOG

Zachary Kessin
http://mostlyerlang.com
@zkessin
WHAT IS QUICKCHECK?
QUICKCHECK

- Don’t write tests: Generate them
- Express invariance as general properties
- Run Lots of Tests
- Find Error conditions
- Shrink them to smallest cases
Testing systems with State

- MODEL SYSTEM
- CREATE EVENT STREAM
- VALIDATE
WE NEED A MODEL
CREATING A MODEL IN ERLANG

- Sometimes awkward
- You want to not reuse the code
USE AN EXTERNAL MODEL!
WHAT MIGHT MAKE THIS EASIER?
PROLOG?
Seven Languages in Seven Weeks

A Pragmatic Guide to Learning Programming Languages

Bruce A. Tate

Edited by Jacquelyn Carter
WHAT IS PROLOG?
PROGRAMMING WITH LOGIC
VERSIONS OF PROLOG

SWI Prolog

Gnu Prolog

ECLiPSe

xsb

Erlog

YAP rolog
HISTORY OF PROLOG

- Invented in the Early 1970’s
- Often used in AI
- Used to Develop Erlang
- What is, WATSON?
Prolog interpreter in and for Erlang

rvirding / erlog

40 commits  2 branches  0 releases  4 contributors

branch: master  erlog / +

Merge pull request #7 from zkessin/master

rvirding authored on 25 Apr
latest commit 39402c2b4d

- bin
  Create Erlang start file erlog

- doc
  Add memberchk/2 predicate to lists library

- ebin
  Move sort/2 from erlog_int to erlog_lists

- examples
  Export consult_file/2 and reconsult_file/2 from erlog.erl.

- src
  now can supply the goal as a string

- .gitignore
  Remove .beam files from Git.

- Emakefile
  Remove .beam files from Git.

- LICENSE
  Convert to use Apache License, Version 2.0

- Makefile
  Improve Makefile to do conditional compilation

0 comments

SSH clone URL

You can clone with HTTPS, SSH, or Subversion.

Clone in Desktop

Download ZIP
ERLOG FEATURES

- ISO Prolog
- Runs in an Erlang Process
- Can talk to ETS tables
- Understands Erlang Data
FACTS

1. model("KEY","value").
2. edge(a,b).
3. edge(b,c).
SIMPLE RULE

1. `connected(A, B) :-`
2. `edge(A, B);`
3. `edge(B, A).`
UNIFICATION

1 edge(a,b).
2 edge(a,c).
3 edge(c,d).
4 edge(e,f).
5 edge(f,b).
6
7 connected(A,B) :-
8     edge(A,B);
9     edge(B,A).
UNIFICATION

1 ?- connected(a,X).
2 X = b ;
3 X = c ;
REVERSIBLE

1 ?- connected(X,f).
2 X = e ;
3 X = b.
RECURSIVE RULES

1 path(A,B,Path) :-
2      travel(A,B,[A],Q),
3      reverse(Q,Path).
4
5 travel(A,B,P,[B|P]) :-
6      connected(A,B).
7 travel(A,B,Visited,Path) :-
8      connected(A,C),
9      C \== B,
10     \+member(C,Visited),
11     travel(C,B,[C|Visited],Path).
QUERIES

1  ?- path(a,f, Path)
2  Path = ...
STATE IN PROLOG

1 assert(FACT).
2 asserta(FACT).
3 assertz(FACT).
4 retract(FACT).
WHAT TYPES OF PROBLEMS?

- State Model
- Algebra
- Grammar
- FSM
MODEL: LAST VALUE CACHE
LVC MODEL IN PROLOG

1 add_to_model(Key, Value) :-
2    retract(model(Key, _)),
3    asserta(model(Key, Value)).
SETTING UP PROLOG TO RUN

1 \{ok,Erlog\} = erlog:start_link(),
2 ok = erlog:consult(Erlog, "../test/erl_cache_model.pl"),
3 erlog:halt(Erlog).

set(Erlog, Key, Value) ->
case erlog:prove(Erlog,
{add_to_model,Key,Value})
of
{succeed, _} -> true;
_R     -> false
end.
CALLING PROLOG

1 \texttt{get}(	exttt{Erlog}, \texttt{Key}) \rightarrow
2 \quad \texttt{PR} = \texttt{erlog:prove}(	exttt{Erlog},
3 \quad \{\texttt{model}, \texttt{Key}, \{\texttt{'Y'}\}\}) ,
4 \quad \texttt{case} \quad \texttt{PR} \quad \texttt{of}
5 \quad \quad \{\texttt{succeed}, [\{\texttt{'Y'}, \texttt{Value}\}]\} \rightarrow
6 \quad \quad \{\texttt{ok}, \texttt{Value}\};
7 \quad \quad \texttt{fail} \quad \rightarrow
8 \quad \quad \texttt{not\_found}
9 \quad \texttt{end}.
ALGEBRA

- Sets
- Graphs
- Trees
- CRDTs
PARTIALLY ORDERED SETS
ESTABLISHING PATHS

1 path(A,B,Path) :-
2     travel(A,B,[A],Q),
3     reverse(Q,Path).

```prolog
1 path(A,B,Path) :-
2     travel(A,B,[A],Q),
3     reverse(Q,Path).
```
ESTABLISHING PATHS

1 travel(A,B,P,[B|P]) :-
2    connected(A,B).
3 travel(A,B,Visited,Path) :-
4    connected(A,C),
5    C \== B,
6    \+member(C,Visited),
7    travel(C,B,[C|Visited],Path).
HEAD/CHILD

1\texttt{child(A)} :-
2\texttt{\neg\text{edge}(A,\_)}.
SIBLING RELATIONSHIPS

1 \texttt{sib}(A,B) :-
2 \texttt{path}(C,A,\_),
3 \texttt{path}(C,B,\_),
4 \texttt{\textbackslash +path}(A,B,\_),
5 \texttt{\textbackslash +path}(B,A,\_).
ANCESTOR / DESCENDENT

1 ancestor(A,B) :-
2 path(A,B,_).
3
4 descendant(A,B) :-
5 path(B,A,_).
HOW TO TEST WITH IT

- Create an initial state
- Apply Event Stream
- Validate states of VC’s after each step
Learn Prolog Now
http://learnprolognow.org
THANK YOU!

- Zachary Kessin
- @zkessin
- @MostlyErlang
- http://mostlyerlang.com