Property-based Testing of Prolog Predicates

Cláudio Amaral[1,2] Mário Florido[1,2] Vítor Santos Costa[1,3]

[1]Dep. of Computer Science, Faculty of Sciences, University of Porto
[2]LIACC - University of Porto [3]CRACS – University of Porto

June 4, 2014

Test/Property Definition

- Code that evaluates to a truth/boolean value
- Can be parameterised

Executing the code with defined parameters checks if the property holds for that particular test case

Properties as Boolean Functions – `True/False`

- Body represents the test format
- Arguments are test cases
- Validity of a test given by the function's result

Automatic Testing

- Evaluate the function-property for a set of test cases

Which test cases?

Test-case Generators

- Procedures that generate values of a predetermined domain
  - Systematic
  - Pseudo-Random
  - Mixed

Prolog Correspondence

- Boolean function
  - True
  - False
- Arguments

- Goal
  - Succeeds
  - Fails
- Input Parameters

Properties in Prolog

```
app([], YS, YS).
app([X|XS], YS, [X|AS]) :- app(XS, YS, AS).
```

```
prop({appLLL, L1, L2}) :- app(L1, L2, L),(L=[] ; L = [_|_]).
```

Properties in Prolog

prop({appLLL, L1, L2})

Properties in Prolog

```
prop(appL) :-
        for_all(listOf(int), L1,
          for_all(listOf(int), L2,
            prop({appLLL, L1, L2}) ) ).
```

Properties in Prolog

```
prop({appLLLen, L1, L2}) :-
    app(L1,L2,L), length(L1, K1),
    length(L2, K2), length(L, K), K is K1 + K2.
```

Properties in Prolog

prop({appLLLen, L1, L2})

Properties in Prolog

```
prop(appLen) :-
        for_all(listOf(int), L1,
          for_all(listOf(int), L2,
             prop({appLLLen, L1, L2}) ) ).
```

Generators and Generator Predicates

- Generators
  - int
  - listOf (ElemGen)
- Generator Predicates
  - int (Int, Size)
  - listOf (ElemGen, List, Size)

A Generator Predicate

```
listOf (Gen, List, Size) :-
      choose(0, Size, K, Size),
      vectorOf(K, GenA, AS, Size).

vectorOf(0, _GenA, [], _Size) :- !.
vectorOf(K, GenA, [A|AS], Size) :-
      call(GenA, A, Size),
      K1 is K-1,
      vectorOf(K1, GenA, AS, Size).
```

Specifying Predicates

Through predicate specification clauses

- Types
- Domain

Specifying Predicates

```
non_empty( [[_|_],_,_] ).
non_empty( [_,[_|_],_] ).
```

{app,3b}

```
 of_type   (listOf(int),   listOf(int),   variable)

    such_that   m:non_empty.
```

Specifying Predicates

Through predicate specification clauses

- Types
- Domain
- Modes
- Answers

Specifying Predicates

{app, 4b}

of_type (listOf(int), variable, variable)

where (i(g, v, v), o(g, v, ngv), o(g, v, v))

has_range {1,1}.

Specifying Predicates

Through predicate specification clauses

- Types
- Domain
- Modes
- Answers
- General post condition

Specifying Predicates

{app, 5}

of_type
(A-(listOf(int)), B-(listOf(int)), C-(variable))

  post_cond
(length(A,K1), length(B,K2), length(C,K),
   K is K1+K2).

Thank you

Specification Clause

{Predicate, Id} of_type (T1, …, Tn)

such_that relation %% ([A1,...,An])

where i(Mi1,...,Min), [o(Mo1,...,Mon)]

has_range {Min,Max} limit K

post_cond Goal

.

Specifying Predicates

```
{app, 1}
 of_type (A-(listOf(int)),  B-(variable),  C-(variable))
 where (i(g, v, v), o(g, v, ngv))
has_range {1,1}.
```

```
>  out   o(g,v,ngv),   [[],_10258,_10258]
```

Composing Properties

- Prop1 and Prop2
- Prop1 or Prop2
- if Cond then Prop1 else Prop2
- for_all(Gen, Var, Prop)
- prop(Label)