Tips On Using Ladybug

Craig A. Damon
cdamon@cs.cmu.edu

Constructing The Specification

A good Z spec may not be a checkable spec

Checkable specs contain

- System descriptions
- Operations
- Desirable properties
- Claims

Claims take two basic forms

- System => Desirable
- Desirable and Operation => Desirable'

  or

- Desirable and Op1;Op2;Op3 => Desirable'

Separate Requirements From Desirable

Crucial: separate requirements / desirable properties

Requirements describe what implementation must do

Desirable properties are behaviors to demonstrate

Need to decide what will be checked first
    Better to err on side of separation

Language Constructs

Ladybug input language NP is subset of Z

Purely first order

No quantifiers

No numbers

Type Inferencing

Ladybug needs to assign types to all expressions

Generic elements can be problematic

- Un
- Id

Ladybug may complain about lack of type info

Add type info with domain/range restriction

- Type <: Un :> Type2

Quantifiers

NP is quantifier-free

All variables implicitly universally quantified

Counterexamples solutions to negations

Negation swaps universal/existential

$\neg \forall x, y. (f(x, y) \Rightarrow g(x, y))$

$\exists x, y. (f(x, y) \wedge \neg g(x, y))$

"Extra" variables are existential or universal

Based on placement

$\exists x, y. (f(x, y, \exists z) \wedge \neg g(x, y))$

or

$\exists x, y. (f(x, y) \wedge \neg g(x, y, \forall z))$

Cross Product

No support for set1 $\times$ set2

Simple replacement

set1 <: Un :> set2

Using Projection Functions

No support for type $\mathbb{P}(\text{Type1} \times \text{Type2})$

Emulate most behavior with projection functions

field1Func : MapType -> Type1

field2Func : MapType -> Type2

Needed to add new given type (MapType)

Limits tractable size

Hard to constrain interactions

$(\text{Type1} \times \text{Type2}) \rightarrow \text{Type3}$

Mappings may repeat pairs

Loading the Specification

Error messages identify parse/type errors
- Error messages repeat line with error
- Gives line number.offset at start of message

Reload button is a big help when fixing spec problems
- Loses scope settings (fixed in next major release)

Ladybug Restrictions

Largest tractable problems are about 10^{80} cases

6x6 relation 10^{11} cases by itself

Largest viable scope is about 8

Maximum 31 enumerated variables (ignoring derived)

Up to 64 in upcoming release

Choosing The Scope

- Consider scope carefully
  - Too big - analysis not tractable
  - Too small - misses bugs
- Consider interactions
  - Projection map types
  - Cross product types
  - Injective or functional
- Consider distinct roles
  - Scalar variables
  - Overlapping sets

Check For Liveness

Easy to over specify

Check for liveness

- Select schema (or operation)
- Check returns example not counterexample

State properties

- Check that at least some states exist

Operations preserving properties

- Check that executions of operations exist

Running Ladybug

Limit options allows multiple counterexamples/executions

Progress meter gives sense of progress
- Progress may not be steady
- Algorithmic flaw over reports early portions

Display/More Options ; Display/Num Values
- Reports number values per variable
- Helps check problems when no cases considered

Display/More Options ; Display/Computed Types
- Can help show isolated segments of the problem
- Variables with distinct types have no overlap