frama C
Software Analyzers

Advanced ACSL and WP tutorial

Virgile Prevosto

April 25th, 2013

frama C
Software Analyzers

Basic contract

- **`requires`**
- **`ensures`**
- **`assigns`**
- validity of pointers
- bounds of arithmetic values

Behaviors

- Specification by cases
- the subdomain is defined by **`assumes`** clause
- can give additional constraints with local **`requires`** clauses
- the behavior's postcondition is defined by **`ensures`**, **`assigns`** clauses
- **`complete behaviors`** states that given behaviors cover all cases
- **`disjoint behaviors`** states that given behaviors do not overlap

Loop invariants - some hints

How to find a suitable loop invariant? Consider two aspects:

- identify locations modified in the loop
  - define their possible value intervals (relationships) after k iterations
  - use **`loop assigns`** clause to list variables that (might) have been assigned so far after k iterations
- identify realized actions, or properties already ensured by the loop
  - what part of the job already realized after k iterations?
  - what part of the expected loop results already ensured after k iterations?
  - why the next iteration can proceed as it does? ...

A stronger property on each iteration may be required to prove the final result of the loop.

Logic functions and predicates

- can be defined directly or through **axiom**s
- may be parameterized by one (or more) program states:
  ```
  predicate foo{L}(int* a) = \at(*a,L) == 0;
  ```
- `\at(\cdot,L)` can be omitted if there is no ambiguity (exactly one state in context).
  - this is what was done until now
- can be completed by additional **lemma**s

Separation

`\separated(loc_1,loc_2,...,loc_n)` expresses the fact that `loc_1`, `loc_2`, `loc_n` are disjoint blocks of memory.

code annotations

- Can be specialized for a given behavior:
  `for b1,b2: assert \true;`
- Will be proved under the **assumes** of the behavior(s)
- But will also only be used for the ensures of the behavior(s)