FORMAL METHODS FOR EXTENSIONS TO CAS

Martin Dunstan, Tom Kelsey, Ursula Martin & Steve Linton

School of Computer Science
University of St Andrews
{mnd,tom,um,sal}@dcs.st-and.ac.uk

September 21, 2000

INTRODUCTION

Computer Algebra Development

Problems

$$\int_{x=0}^{\infty} \frac{dx}{4x^4+1} = 0 \quad \text{in AXIOM}$$

$$\int_{0}^{\infty} \frac{\cos x}{x^2+1} dx \in \mathbb{C} \quad \text{in Maple}$$

Designers

- Type system & default methods
- Sound mathematical algorithms

Library Developers

- Are the type system and methods unambiguous?
- Are the restrictions on algorithms explicit?

Lightweight Formal Methods

Lightweight?

- Jackson and Wing, *IEEE Computer* 1996
- Replace provable correctness of system by an emphasis on the reduction (if not the elimination) of design and implementation errors.

Applicability to CAS

- Parts of CAS are formal enough
- Verification of maths code can be non-trivial
- Developers need precise definitions and conditions for use

Aims

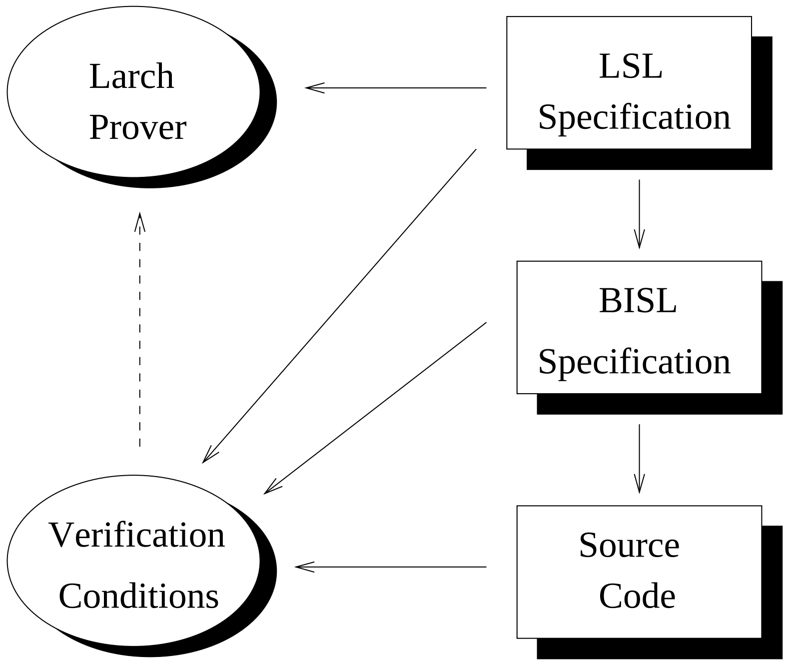
Larch and AXIOM

- Larch is two-tiered: abstract and interface
- AXIOM is two-tiered: category and domain
- LSL specification of type hierarchy
- Larch/Aldor specification of new code

Benefits

- Unambiguous definition of primitives
- LP proofs of abstract properties
- Automatic generation of Verification Conditions
- LP available to discharge VCs

Development Diagram



Abstract Algebra

Commutative Ring

- Additive abelian group - $a + b = b + a$, $a + 0 = a$
- Multiplicative abelian monoid - $a * b = b * a$, $a * 1 = a$
- Multiplication distributes over addition
  $a * (b + c) = a * b + a * c$
- Example - polynomials over $\mathbb{Q}$

Integral Domain

- Commutative ring with no zero divisors
- Two non-zeros multiply to a non-zero
- Example - the integers $\mathbb{Z}$

Field

- Integral domain with multiplicative inverses
- If $a \neq 0$, then $\exists\, b$ such that $a * b = 1$
- Examples - the reals $\mathbb{R}$ and the rationals $\mathbb{Q}$

CASE STUDY

Motivation

- Given side-conditions for AXIOM types
- Conditions are informal comments which can be inaccurate

```
ComplexCategory(R:CommutativeRing):
    ⋮      ⋮      ⋮
if R has IntegralDomain then IntegralDomain
if R has Field then Field -- this is a lie; we
must know that x**2+1 is irreducible in R
    ⋮      ⋮      ⋮
```

We know that augmenting a commutative ring with an imaginary element should yield another commutative ring

- The library developer
  - may not be aware of the comments
  - can be misled by the comments

```
ComplexCategory (CR) : trait
assumes CommRingCat (CR)
includes RequirementsForComplex (CR)
introduces
  imaginary, 0, 1 : → T

    ⋮     ⋮     ⋮

asserts ∀ w,z : T
  imaginary == comp(0,1);
  0 == comp(0,0);
  1 == comp(1,0);

    ⋮     ⋮     ⋮

implies
   AbelianGroup(T,+),            ⎫
   AbelianMonoid(T,*)            ⎬ A
   Distributive(+,*,T),          ⎭
  ∀ z,w : T
    imaginary*imaginary == -1;   } B
```

A: Commutative ring in gives commutative ring out

B: Check on basic property of `imaginary`

```
TypeConditions (CR,T) : trait
includes
  CommRingCat (CR), ComplexCategory (CR)
introduces
  TC1, TC2, invsExist : → Bool
asserts ∀ a,b,c : CR
  TC1 ⇒ (a ¬= 0 ⇒ a*a ¬= -(b*b));
  TC2 ⇒ (a*a ¬= -1);
  invsExist ⇒ (a ¬= 0 ⇒ ∃ c (a*c = 1))
implies ∀ v,z,w : T
  TC1 ∧ nZD ∧ invsExist
          ⇒ (w ¬= 0 ⇒ ∃ v (w*v = 1));            } A
  TC2 ∧ nZD ∧ invsExist
          ⇒  (w*z=0 ⇒ w=0 ∨ z=0);                } B
  TC1 ∧ nZD ⇒ (w*z=0 ⇒ w=0 ∨ z=0)                } C
```

If input is:

A: a field with `TC1`, then output is a field

B: a field with `TC2`, then output is an integral domain

C: an integral domain with `TC1`, then output is an integral domain

Interface Approach

Same problem in a different, yet complimentary, way

1. Define the functor `Complex` in Larch/Aldor

```
++} requires isIntDomain(CR)
 ∧  ¬∃ x,y:CR • (x ≠ 0 ⇒ x*x + y*y = 0);
++} ensures  isIntDomain(%);
++} modifies nothing;
Complex(CR:CommutativeRing):CommutativeRing;
```

2. Instantiation with `Int` generates the VC

   $\texttt{isIntDomain}(\texttt{Int}) \wedge \neg\exists x, y : \texttt{Int} \bullet (x \neq 0 \Rightarrow x^2 + y^2 = 0)$

3. We obtain the useful post-condition

$$\texttt{isIntDomain}(\texttt{Complex}(\texttt{Int}))$$

4. Instantiation with `PrimeField 5` generates the VC

$$\neg\exists x, y : \texttt{PrimeField5} \bullet (x \neq 0 \Rightarrow x^2 + y^2 = 0)$$

   which can be proved false ($x = 2$, $y = 4$)

CONCLUSIONS

LSL Specifications

- Exist for every algebraic AXIOM category
- Exist for AXIOM functors (`Fraction`, `Complex`, $\cdots$)
- Refined using textbook properties (e.g. prove, in LP, the quotient rule in the theory of `DifferentialRing`)
- Provide well defined primitives and conditions for use at the interface level
- Highlight areas in which computational maths differs from abstract maths
- Can be used as a formal basis for other CAS implementations

Interface Specification

Larch/Aldor

- Formal notation for describing AXIOM/Aldor behaviour
- Allows Larch annotations to Aldor code to be recognised
- Provides mechanism for generating VCs

VCs

- Many discharge automatically

$$\texttt{isIntDomain}(\texttt{PrimeField}\ \ 5)$$

- Others are more interesting

$$\texttt{isOdd}(\texttt{Order}\ \ G : \texttt{Group}) \implies \texttt{isSoluble}\ \ G$$

- Aid compiler optimisation and method selection
- VC generation (ideally) happens in the compiler