Dependable Scientific Computing

Gabriel Dos Reis

Parasol Lab
Department of Computer Science and Engineering
Texas A&M University
http://parasol.tamu.edu/~gdr/

May 1, 2009
Outline

Programming Tool “Builder”

The Dependable Scientific Computing Challenge
Outline

Programming Tool “Builder”

The Dependable Scientific Computing Challenge
Current Research Work

- **Programming Languages**
  - Extending ISO C++ standards: C++0x

- **Compiler Technologies**
  - Collaborative work on static analysis, semantics-based transformation framework of ISO C++ programs

- **Scientific Computing**
  - OpenAxiom
  - IEEE Interval Arithmetic standard
Why Work on C++?

- Widely used in scientific computing community
- Available on virtual any computing platform
- Efficient, expressive, support sound programming methodologies
- Has an ISO standard definition
- ...
Extending ISO C++ standards

- Improved support for Generic Programming
  - C++ concepts
  - local type inference
  - generic type aliasing
- Improved support for system programming
  - generalized constant expressions
  - ...
- More regular language rules
  - generalized initializer list
  - ...
- ...

- ...
namespace algebra {
    // A simplified version of the ‘monoid’ structure.
    concept Monoid<Regular T> {
        // require existence of a binary operation
        T operator+(T, T);
        // and a way to retrieve the neutral value
        T neutral_value();

        // semantics properties -- for use by optimizer
        // or static analysis tools. The parameter
        // ‘x’ really stands for a computation, e.g. AST.
        axiom identity(T x):
            x + neutral_value() <=> x;
            neutral_value() + x <=> x;
        // associativity of the + operation
        axiom associative(T x, T y, T z):
            (x + y) + z <=> x + (y + z);
    }
}
namespace algebra {
    // for all containers C such that C::value_type
    // is a regular type
    template<Container C>
        requires Monoid<C::value_type>
    C::value_type reduce(const C& c)
    {
        // use T as short-hand for C::value_type
        using T = C::value_type;
        // initialize result of the left reduction
        auto result = T::neutral_value();

        // accumulate all values from the container
        for (v: c)
            result = result + v
        return result;
    }
}
// the standard string datatype is also a monoid
concept_map algebra::Monoid<string> {
   // we don’t need to define operator+, because the
   // library operator function
   // 'operator+(const std::string&, const std::string&)'
   // meets the syntactic type requirements.

   string neutral_value() { return ""; } }

int main() {
   vector<string> cat = { "Category", " ", "Theory" }; 
   return algebra::reduce(cat).size();
}
C++ Concepts: my biased view

- 1990s: Seminal work of Alex Stepanov et al. (STL notably)
- personal exposure to AXIOM, Aldor during FRISCO.
- September 2008: First draft of C++0x formally approved, Summit, New Jersey.
Outline

Programming Tool “Builder”

The Dependable Scientific Computing Challenge
Some Future Directions for Symbolic Computation

- Increase the communication channels with other disciplines
  - Programming Language community
  - Compiler Construction community
  - Theorem Proving community
  - ...

- Machine checked/verified algebraic and symbolic computation algorithms and systems
  - A POPLmark challenge for ISSAC community?

- Lead the way to intelligent systems
  - Propose problems for PL researchers and tool builders
Shameless plug

2009 ACM SIGPLAN Workshop on Programming Languages for Mechanized Mathematics Systems Munich, Germany; August 21, 2009

Deadlines: May 11 (abstract), May 18 (full paper)

http://plmms09.cs.tamu.edu/
What is dependable software?
- one you can depend on — one you can trust
- provides evidence that support claims

Why should we care?
- scientific computing is central to our society
- vulnerable to coordinated or chained failures

Isn’t this challenging for symbolic computation?
- Yes!
- But, we can.
How?  
# Require that an algorithm implementation provides evidence for claimed specification

Isn’t this overkill?  
# It can be – but it does not need to

Does it scale?  
# still a research field  
# Symbolic computation can contribute and benefit from it

Has this been tried?  
# Yes — chips designers/manufacturers (AMD, Intel, etc.)  
# gaining traction in the PL community (POPLmark challenge)  
# ...

Dependable Scientific Computing
Thanks!