An Automatic Parallelization Framework for OpenAxiom

Yue Li and Gabriel Dos Reis
Texas A&M University
College Station, USA, 77843-3112
{yli,gdr}@cse.tamu.edu

1 Introduction

This poster illustrates an automatic parallelization framework for the OpenAxiom [4] computer algebra system. The objective is to help incidental users or non-expert algebraic library authors benefit from implicit parallelization present in structured algebraic computations. The framework rewrites reductions in algebraic libraries with their parallel versions. For instance, the planar curve formed by a given list of points $(x_1, y_1), \ldots, (x_n, y_n)$ may be approximated with the Lagrange polynomial:

$$P = \sum_{i=1}^{n} y_i p_i \quad \text{where} \quad p_i = \prod_{1 \le j \le n, p \ne i} \frac{X - x_j}{x_i - x_j}$$

Each polynomial p_j can be computed using sequential nested loops. Our framework is able to transform the sequential computation to its parallel version. The transformed version computes the denominator and numerator of each term p_i using a parallel reduction function, respectively. The list of the terms $(X - x_j)$ in the numerator, and the terms $(x_i - x_j)$ in the denominator are computed via calling a parallel map function. The transformation relies on the fact that the multiplication operator over any field and the multiplication operator over univariate polynomials are monoid operators, *i.e.*, the operators are associative and each has an identity element. In general, such algebraic properties are domain-specific knowledge and are difficult to infer systematically. This framework provides linguistic support to express them directly in code:

```
forall(F:Field,P:UnivariatePolynomialCategory(F))
  assume MonoidOperator(P, *) with
    neutralValue = 1$P
```

```
forall(F: Field)
  assume MonoidOperator(F, *) with
    neutralValue = 1$F
```

The name `MonoidOperator` designates the category capturing the algebraic properties for monoid operators:

```
MonoidOperator(T: BasicType, op: (T, T) -> T): Category
  == AssociativeOperator(T, op)
       with neutralValue: T
```

The name `neutralValue` is a `T`-dependent constant denoting the identity element of a monoid operator. The category `AssociativeOperator` defines the rules for associative operators. This shows that properties can be composed out of existing ones.

2 An overview of the framework

The framework is implemented as an OpenAxiom library. The work flow of the framework is Figure 1. The inputs include source code and user written assumptions. A semantics-based static analysis is performed on the source code to identify potential reductions. In general, a reduction is written either as a loop, or

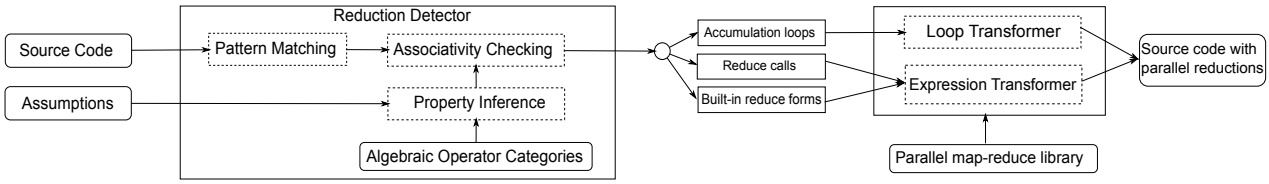

Figure 1: The workflow of the automatic parallelization framework.

a reduction function call. For each potential reduction, the monoid properties of the candidate operator is checked against user assumptions and other previously derived facts. The reductions which pass the property checking are handed to a transformer. The transformer rewrites reductions with their parallel versions provided by a parallel library.

3 Results

The framework has discovered rich parallelization opportunities implied by reductions in OpenAxiom algebra library [1]. With the framework, we parallelized a software installation test, a set of OpenAxiom algebra library functions, and a user application on homotopy continuation method [2]. The experiments are run using a dual-core PC and a cluster. Results show that the framework speeds up the software installation test by 15%, the speed-up varies for different algebra library functions, and up to 5 times speed-up is obtained for the user application. In addition to parallelizing iterative reductions, we also provide a prototype implementation for parallelizing recursive reductions. A recursive reduction is first transformed to its iterative version using the incrementalization based program transformation technique [3]. The iterative reduction is further transformed to its parallel version using the transformation algorithms in the current framework.

4 Acknowledgements

The authors thank the Texas A&M University Brazos HPC cluster for providing computing resources to support the research reported here. This work was partially supported by NSF grant CCF-1035058.

References

[1] Y. Li and G. Dos Reis. A quantitative study of reductions in algebraic libraries. In *PASCO '10: Proceedings of the 4th International Workshop on Parallel and Symbolic Computation*, pages 98–104, New York, NY, USA, 2010. ACM.

[2] Y. Li and G. Dos Reis. An automatic parallelization framework for algebraic computation systems. In *ISSAC '11: Proceedings of the 36th International Symposium on Symbolic and Algebraic Computation*, 2011. to appear.

[3] Y. A. Liu and S. D. Stoller. From recursion to iteration: what are the optimizations? In *Proceedings of the 2000 ACM SIGPLAN workshop on Partial evaluation and semantics-based program manipulation*, PEPM '00, pages 73–82, New York, NY, USA, 1999. ACM.

[4] OpenAxiom. `http://www.open-axiom.org`, 2011.