

Verifying Compiler Generation For A Domain Specific Language

Amala Vijaya Selvi Rajan
Research Student/Tutor
School of computing
Middlesex University
London, UK
a.rajan@mdx.ac.uk

The society is moving away from central mainframe based computing to network based computing to achieve increased performance, sharing of resources, increased extendibility and reliability, availability and fault tolerance[1]. Developing effective distributed systems which are portable, simple, expressive and secured are difficult. Verifying the correctness of such systems is far more challenging.

A Domain specific language is a high level language designed to solve problems in a particular domain and it has been suggested as a mean to develop reliable software [3]. LIPS, Language for Implementation Parallel/Distributing Systems, is originally conceptualised by A.S.Bavan and E.Illingworth [2]. It is a domain specific, parallel/Distributed message passing language, which allows programmers to write code using global view that describes their algorithm at a higher level rather than implementing per-processor behaviour. However, unlike other global view languages, LIPS permits users to reason about the parallel implementation of their code at the syntactic level, allowing them to make informed algorithmic decisions based on the program's parallel implementation.

Hoare says 'a verifying compiler is one that proves mechanically that a program is correct before allowing it to be run' [4]. The main objective of our research is in developing a paradigm for proving the compiler correctness of LIPS. The intended result of this research would be:

- An Unverified compiler using Lex, Yacc, BEG, etc., through which verifying compiler can be constructed;
- Methods/tools which could be used for verification and
- Verifying Compiler itself.

The unverified compiler for LIPS has been constructed using JFlex [5], which is a lexical analyzer generator (also known as scanner generator) for Java, written in Java and CUP [6], which is a Java based Constructor of Useful Parsers. Combination of one or more specification languages is well suited for the specification of concurrent or distributed systems, where both the modelling of processes and state is necessary. Formal verification of LIPS will be done using the specifications of Object-Z [7], and CSP[8].

Keywords: distributed systems, parallel implementation, compiler, lexical analyzer, parser, verifying compiler, formalisms, asynchronous, domain specific language, jflex, cup

REFERENCES:

- [1] W. Gropp, E. Lusk & A. Skjellum, 'Using MPI - Portable Parallel Programming with the Message Passing Interface', MIT Press 1994
- [2] A.S.Bava, E.Illingworth, 'A Language for Implementing Parallel and Distributed systems using asynchronous point-to-point communication.
- [3] G.I Gupta, E. Pontelli, 'Specification, Implementation and verification of Domain Specification Languages: A logic programming based approach', Computat. Logic (Kowalski Festschrift), LNAI 2407, pp. 211–239, Springer-Verlag Berlin Heidelberg, 2002.
- [4] T. Hoare, 'Towards verifying compiler', O. Owe et al. (Eds.): From OO to FM (Dahl Festschrift), LNCS 2635, pp. 124–136, 2004, c @ Springer-Verlag Berlin Heidelberg 2004
- [5] <http://jflex.de/>, 'JFLEX: The fast scanner generator of JAVA', accessed on 3/3/2004
- [6] <http://www.cs.princeton.edu/~appel/modern/java/CUP/>, 'CUP: Parser generator for JAVA', Accessed on 3/3/2004
- [7] R. Duke, G. Rose, and G. Smith. Object-Z: A specification language advocated for the description of standards.', Computer Standards and Interfaces, 17:511–533, 1995
- [8] C. Hoare, 'Communicating Sequential Processes', International Series in Computer Science. Prentice-Hall, 1985