Video Walkthrough

go.lbl.gov/sc22-flang-testing

Agile Acceleration of LLVM Flang Support for Fortran 2018 Parallel Programming

Katherine Rasmussen[1], Damian Rouson[1], Najé George[2], Dan Bonachea[1], Hussain Kadhem[1], Brian Friesen[1]

[1]Lawrence Berkeley National Laboratory, [2]San Diego State University

Introduction

Problem

LLVM's Flang Fortran compiler is currently Fortran 95 compliant, and the frontend can parse Fortran 2018. However, Flang does not have a comprehensive 2018 test suite and does not fully implement the static semantics of the 2018 standard.

Solution

Agile software encourages early delivery of working software subject to continual improvement. We are investigating whether agile techniques centered around pair programming and test-driven development (TDD) can help Flang to rapidly progress to Fortran 2018 compliance. Because of the paramount importance of parallelism in high-performance computing, we are focusing on Fortran's parallel features, commonly denoted "Coarray Fortran." We are developing what we believe are the first comprehensive, open-source tests for Fortran 2018 parallel features. We push our compile-time behavior tests to the main LLVM-Project repository. We push our runtime tests for parallel Fortran features to the repository of the Caffeine parallel runtime library that we are concurrently developing.

Objectives

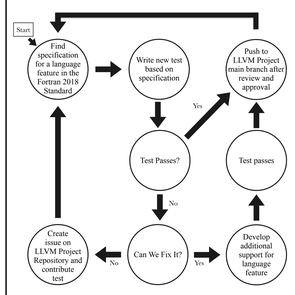
- Exhaustively delineate all of the parallel programming features in Fortran 2018
- Develop semantics tests for LLVM Flang covering statically checkable program errors that the Fortran standard obligates the compiler to detect
- Expand frontend support, including additional error checking, when tests identify missing capabilities

Approach

- Employ agile software development practices
- Test a comprehensive range of standard-conforming and non-conforming Fortran 2018 syntax
- Test-driven development: any contributed tests that fail provide a specification for new features to add to Flang

Agile Development

- Test-driven development



- Pair programming sessions
- Valuable team member interactions
- Get feedback early and frequently
- Leverage existing git and Github tools
- Leverage existing agile practices of the LLVM developer community:
  - Use LLVM's continuous integration (CI) test infrastructure to quickly fix CI failures.
  - Code reviews on Phabricator for feedback, edits, and approvals

GitHub Project Board

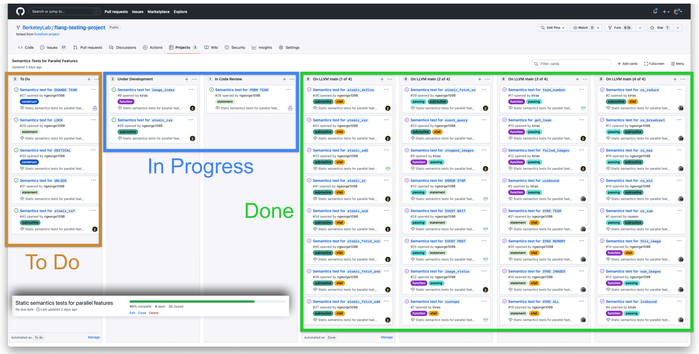


Figure 1: Exhaustive list of Fortran 2018 parallel programming features to test https://go.lbl.gov/flang-testing

Compile-Time Test Coverage

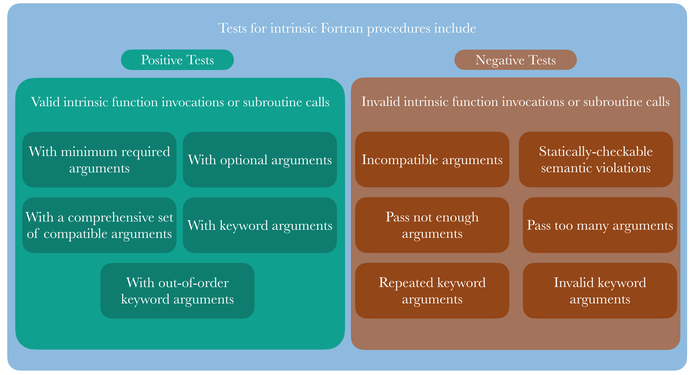


Figure 2: Diagram outlining the components of the static semantic tests for intrinsic functions and intrinsic subroutines

Test-Driven Development Example

CO_SUM (A [, RESULT_IMAGE, STAT, ERRMSG])

Figure 3: Signature for the intrinsic collective subroutine, ***co_sum***, as defined by the Fortran 2018 standard. 'a' is the only required argument and the rest of the arguments are optional.

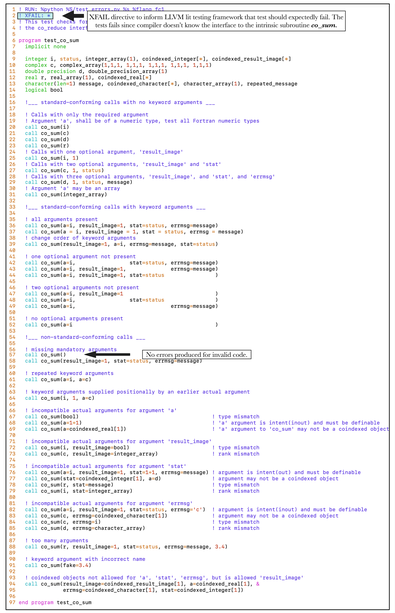


Figure 4: Static semantics test excerpt for the ***co_sum*** subroutine, this test expectedly fails.

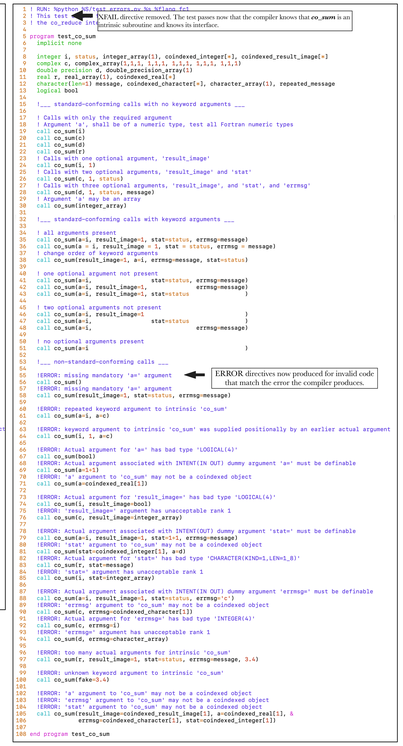


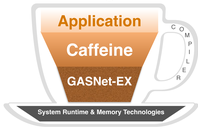
Figure 5: Updated static semantics test excerpt for the ***co_sum*** subroutine that passes after interface is added

```
{"co_sum",
    {{"a",            AnyNumeric,  Rank::anyOrAssumedRank, Optionality::required, common::Intent::InOut},
     {"result_image", AnyInt,      Rank::scalar,           Optionality::optional, common::Intent::In},
     {"stat",         AnyInt,      Rank::scalar,           Optionality::optional, common::Intent::Out},
     {"errmsg",       DefaultChar, Rank::scalar,           Optionality::optional, common::Intent::InOut}},
    {}, Rank::elemental, IntrinsicClass::collectiveSubroutine},
```

Figure 6: Interface to compiler for ***co_sum*** allows the test to pass when combined with a static semantic check for coindexed objects (not shown).

Runtime Tests

- Because Flang cannot yet produce executable files from Fortran 2018 source code, we are developing runtime tests in a separate repository: Caffeine.
- Caffeine is a runtime library that supports parallel Fortran 2018 features.
- Caffeine runs atop the GASNet-EX exascale networking middleware.



go.lbl.gov/caffeine

- For more on Caffeine, see: Rouson & Bonachea (2022) "*Caffeine: CoArray Fortran Framework of Efficient Interfaces to Network Environments*" SC22 Workshop on the LLVM Infrastructure in HPC doi:10.25344/S4459B

Outcomes

- The Berkeley Lab fork of the LLVM-Project GitHub repository includes a project board capturing an exhaustive list of 41 parallel features to test.
- We have pushed static semantics tests for 32 such features upstream to LLVM-Project: intrinsic functions supporting parallelism, collective subroutines, atomic subroutines, synchronization statements, and more.
- We have contributed additional static semantic analysis and error checking for 11 missing parallel features exposed by our tests. More contributions are under development or in code review.
- We contributed error checks for 2 non-parallel features.
- We have developed 44 runtime tests that we exercise by developing Caffeine.

U.S. DEPARTMENT OF ENERGY Office of Science

This research was supported in part by the Sustainable Research Pathways for High Performance Computing (SRP-HPC) a project of the Sustainable Horizons Institute.
This research was supported in part by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration.

ECP EXASCALE COMPUTING PROJECT