Static Analysis of MPI Programs Targeting Parallel Properties

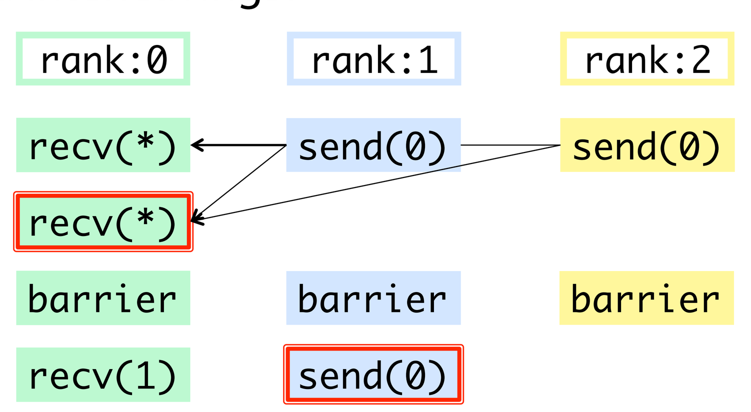
Sriram Aananthakrishnan*, Greg Bronevetsky (advisor)+, Ganesh Gopalakrishnan (advisor)*

*University of Utah +Lawrence Livermore National Laboratory

1. Problem

Parallel properties are important to establish during program analysis/optimization; examples for MPI:

- Does the message exchanged by two MPI operations remain constant?
- Does the communication topology have any collective communication pattern?
- Can `send(0)` match with `recv(*)` under all executions and interleavings?



Compile time detection of parallel properties is efficient. It also enables

- Compiler transformations
- Effective reasoning of MPI applications

2. Challenge

Detecting parallel properties at compile time requires computing communication topology -- **hard!**

All these aspects of MPI communication analysis are unbounded:

- number of processes
- number of MPI operations from loops
- number of paths containing MPI operations
- interleavings due to non-determinism

Statically undecidable!!

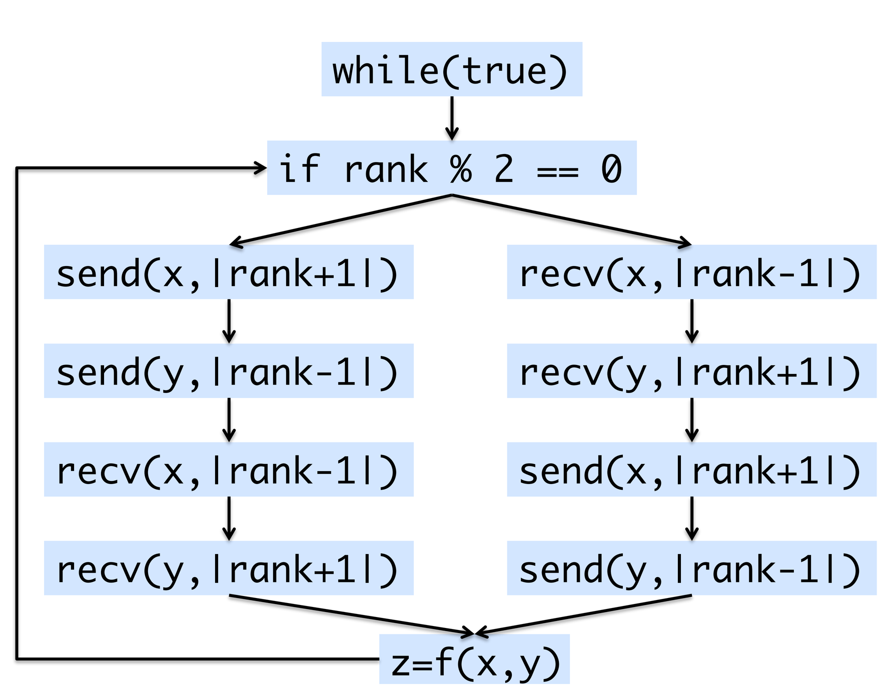


Fig: Control-flow graph of an MPI program with unbounded MPI operations

3. Approach: Approximate Communication Topology

- MPI programs with unbounded number of processes adds complexity in matching MPI operations
- Fix the number of process by modeling MPI program as a CFG cross-product: $CFG_1 x CFG_2 x..CFG_N$
- **Create N analysis instances one for each CFG_i** where each analysis instance is a composition of MPI Value, Constant propagation, Unreachable path, Points-to and MPI Matching dataflow analyses

MPI Value: Assigns concrete values to rank variable corresponding to CFG_i

Constant propagation: Folds the constant rank values to expressions in each CFG_i

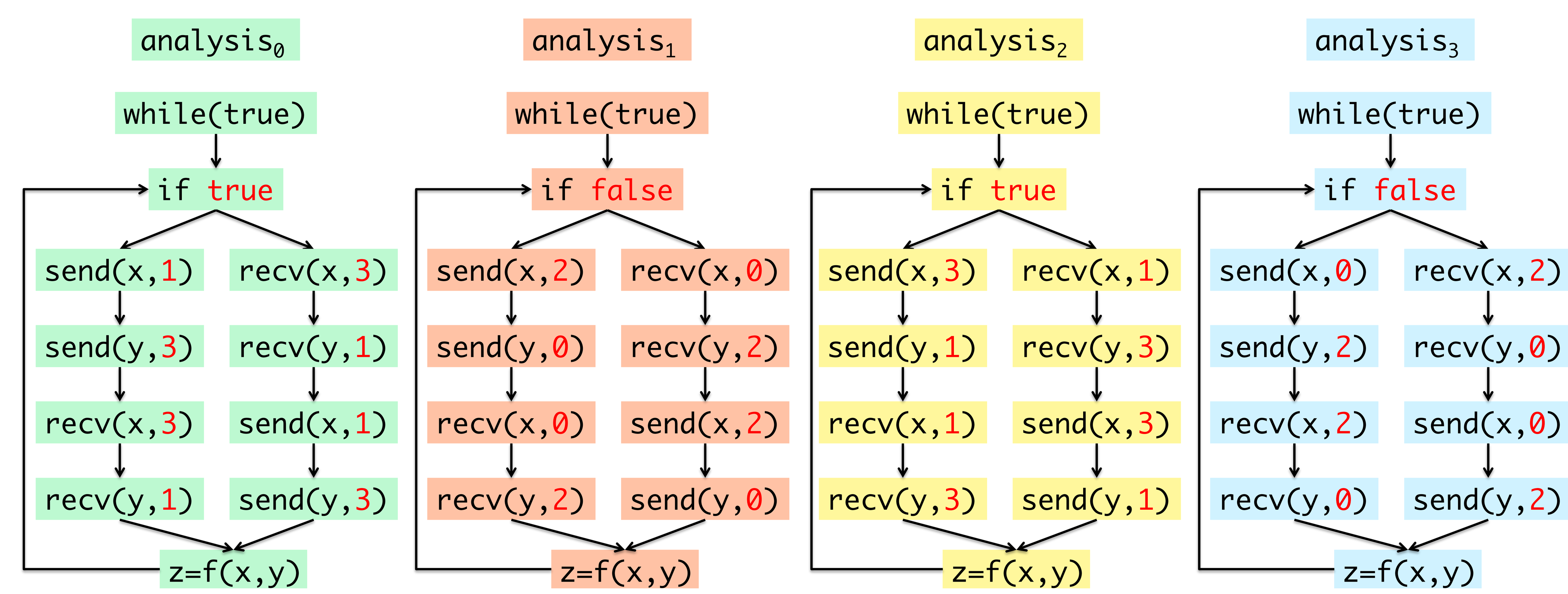


Fig: Analysis instances after MPI Value and Constant propagation

Abstracting MPI Operations: **Each analysis instance groups MPI operations into an equivalence class if they are issued from the same statement.**

Computing abstract communication topology: Match the equivalence classes by forwarding the corresponding operation to MPI runtime. Exchange the dataflow state as the message, which is merged at the receiver using join operator.

Sound approximation: No communication behavior is omitted

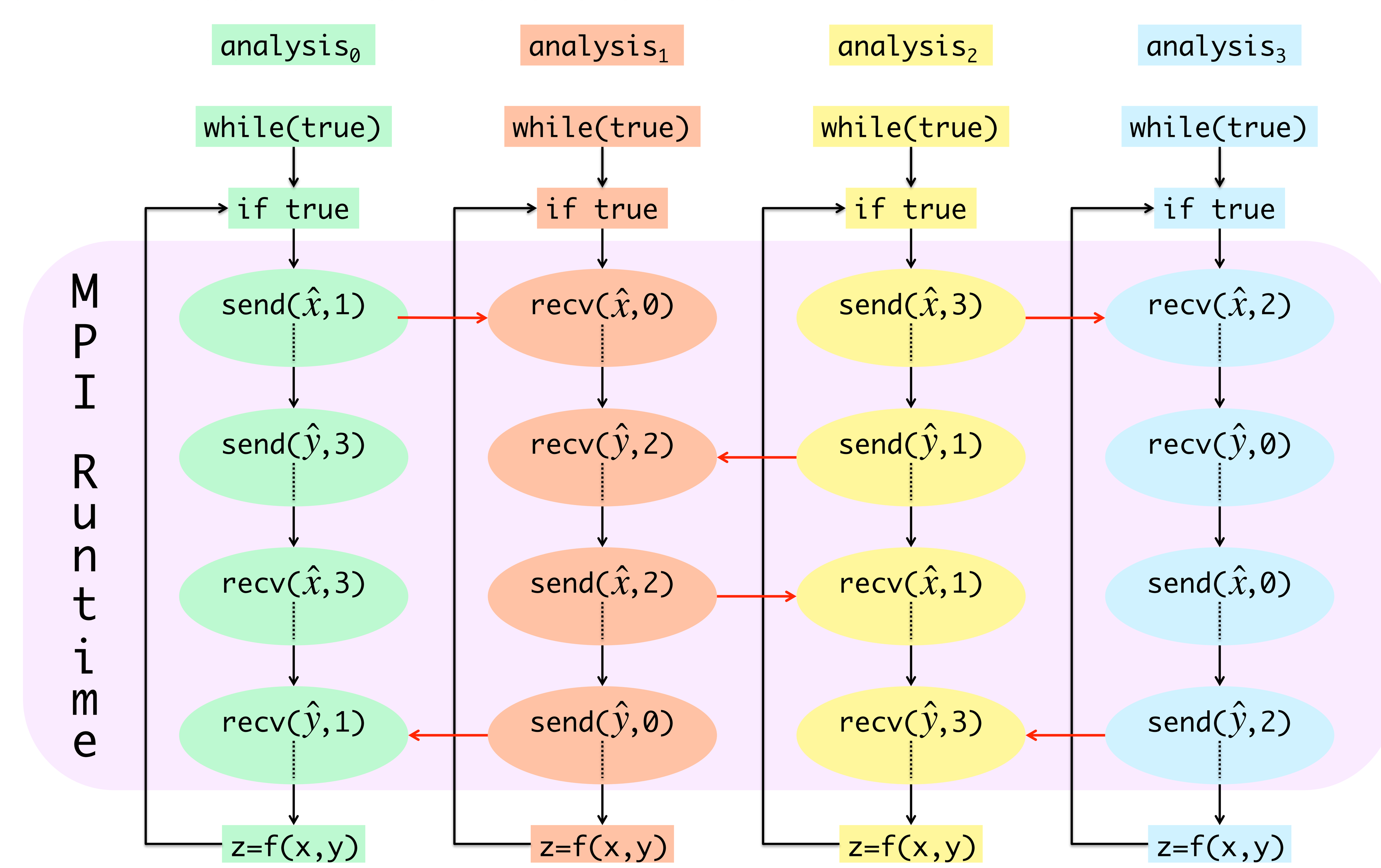


Fig: Analysis instances after unreachable path and MPI matching

4. Implementation Status

Two types of composition techniques implemented

- **Loose:** Analyses are run one after the other sequentially
- **Tight:** Analyses are run in lock-step manner simultaneously improving the precision of each other reducing compositional analysis time
- Analyses interact using FUSE[1] interface
- **Implemented on FUSE framework in ROSE compiler with support for a large set of C++ and MPI primitives**

	MPI Val	CP	UC	PT	MPI Mat
Loose	✓	✓	✓	✓	×
Tight	✓	✓	×	✓	×

Table: Implementation status

5. Previous Work

Strout, Kreaseck, Hovland, Dataflow analysis for MPI Programs, ICPP'06

- Grouped operations based on constraints on target expressions
- Equivalence classes are matched following MPI matching rules

Bronevetsky, Communication-sensitive static dataflow for parallel message passing applications, CGO 2009

- Grouped processes into sets of equivalence classes
- Grouped MPI operations if they are issued from same statement
- Employs complex matching algorithm

Advantages of our approach

- Simplified matching (MPI matching handled separately through dynamic component)
- Dynamic component flows data-flow facts back into composable analysis interface of FUSE
- Scalable parallel implementation

6. References

1. Bronevetsky, Burke, Aananthakrishnan, Zhao, Sarkar "Compositional dataflow via abstract transition systems", LLNL-TR 2013

This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, and was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344 (LLNL-POST-662080)