Native Handling of Message-Passing Communication in Data-Flow Analysis

Valérie Pascual – Laurent Hascoët

AD2012 Fort Collins - July 2012

Outline

- Static data-flow analyses of programs
- Data-flow analyses + Message-passing
- Channels and Flow graph local restart
- Performance discussion
- Choosing a good set of channels
- Implementation
- Further work

Static data-flow analyses of programs

Necessary in Automatic Differentiation for efficient automatic transformation of codes:

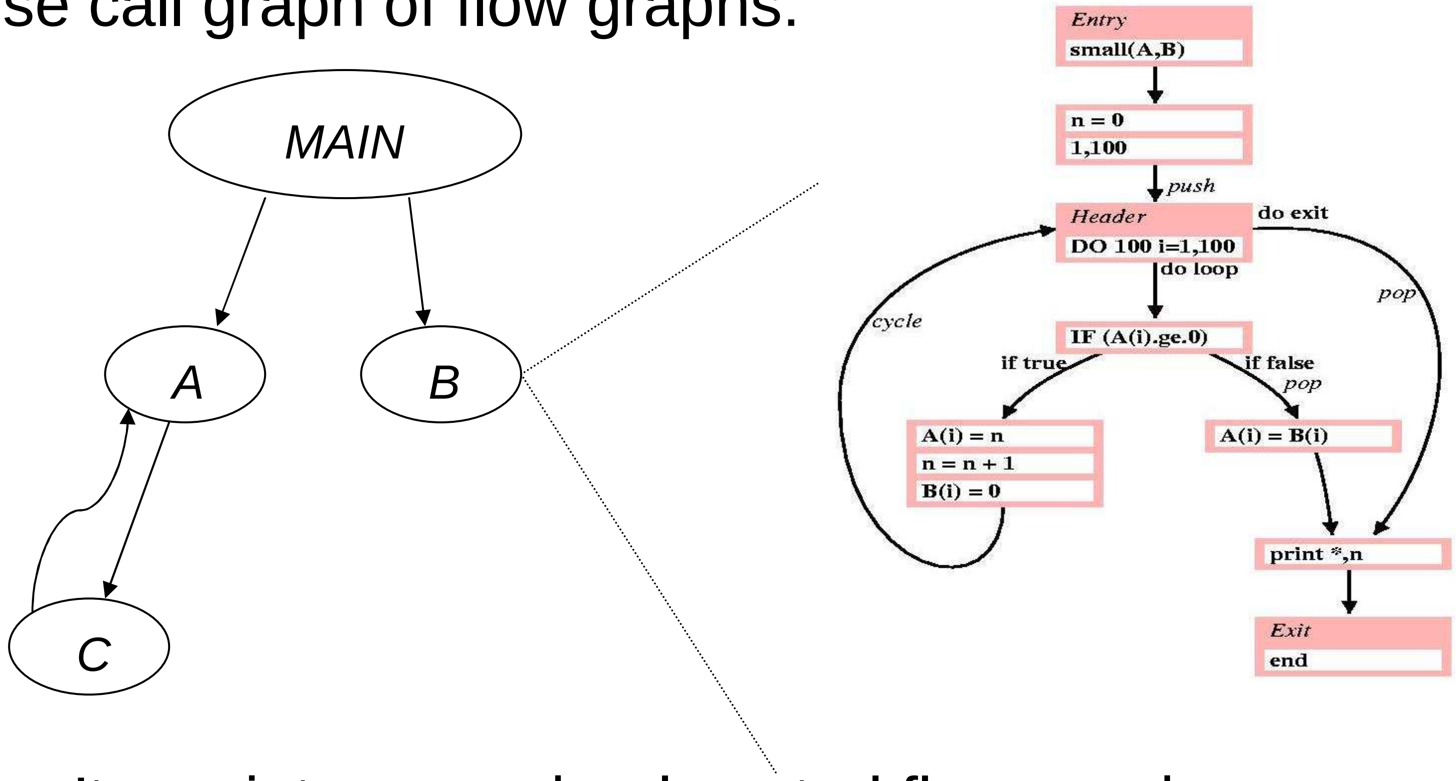
In-Out, Pointer, Activity, Differentiable dependency, Diff liveness, TBR

Static analyses (i.e. at compile-time)

$\Rightarrow$ approximation   generalization

Static data-flow analyses of programs

We use call graph of flow graphs:



We don’t use interprocedural control flow graph

Static data-flow analyses of programs

- Context sensitive on call graphs

  generalization on calls ⇒ summarized information for procedures
- Flow sensitive on flow graph

Both graphs may be cyclic

⇒ Fix-point propagation (using worklists)

Example of data-flow analysis: activity

A variable is active if it is varied and useful:

varied: it depends on an independent input

useful: it influences the depend output

Implemented as three data-flow analyses:

- dependency: bottom-up on the CG, forward on the FG
- varied: top-down on the CG, forward on the FG
- useful: top-down on the CG, backward on the FG

Data-flow analyses + Message-Passing

```
subroutine q(x, y, rank, code, tag, status)
 include 'mpif.h'
 real :: x, y
 integer :: rank, code, tag
 integer, dimension(MPI_STATUS_SIZE) :: status
 if (rank == 0) then
      call MPI_SEND (x, 1, MPI_REAL , 1, tag,
         MPI_COMM_WORLD ,code)
 else
      y = 0
      call MPI_RECV (y,1, MPI_REAL , 0, tag,
            MPI_COMM_WORLD ,code, status)
 end if
end subroutine
```

Data-flow analyses + Message-Passing

New flow of data unrelated to the flow-graph

Propagation algorithm must be extended to capture this

A review of existing answers:

→Use fictitious global communication variables

→Assign analysis’ conservative default value to all variables transmitted through message-passing

→Use augmented interprocedural control-flow graph with special flow arrows that convey messages: MPI-ICFG

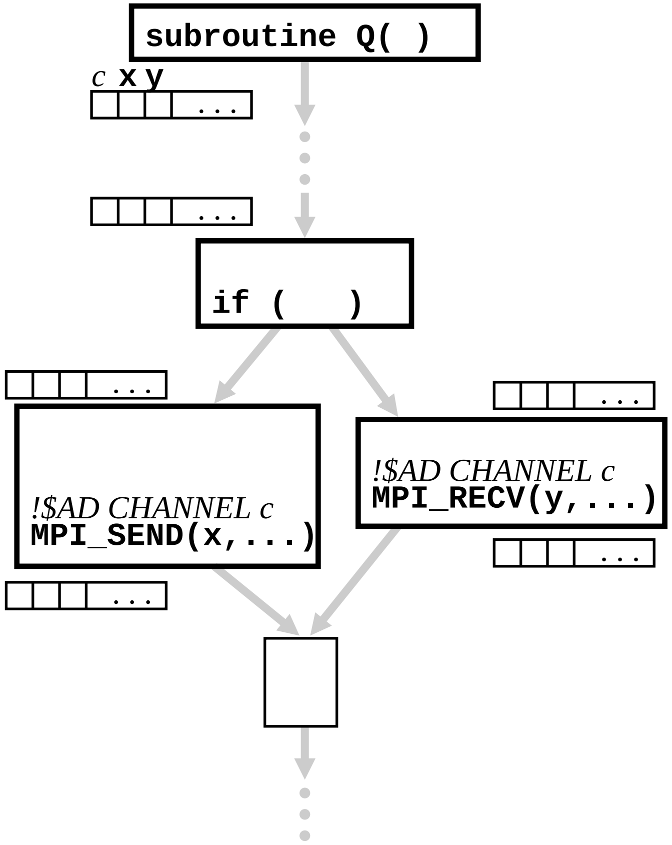
Channels and Flow graph local restart

For any data-flow analysis

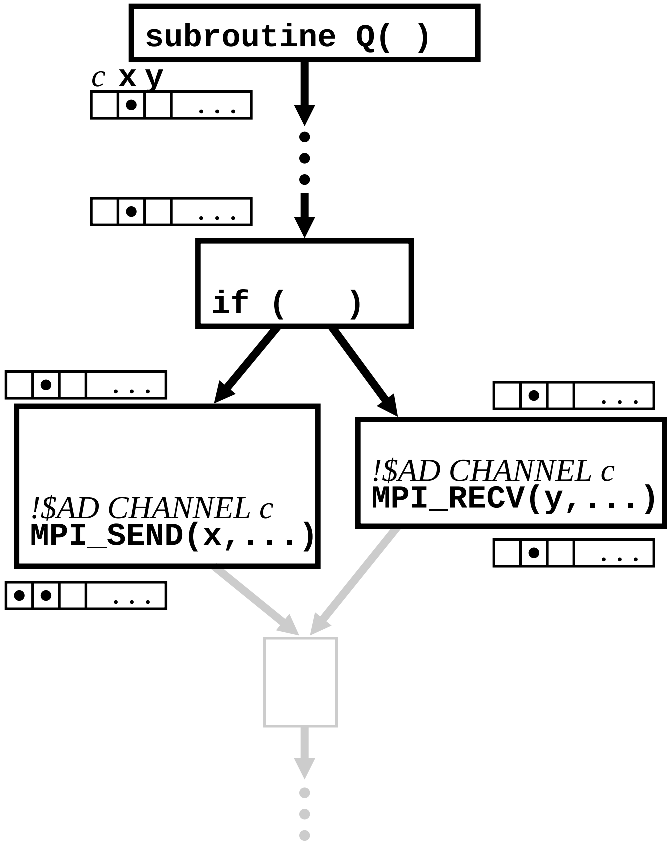
- Introduce new global variables to represent communication channels
- Modify the propagation algorithm on the FG when reaching a communication call

⇒ Restart by adding the entry block (forward) or exit block (backward) on top of the worklist

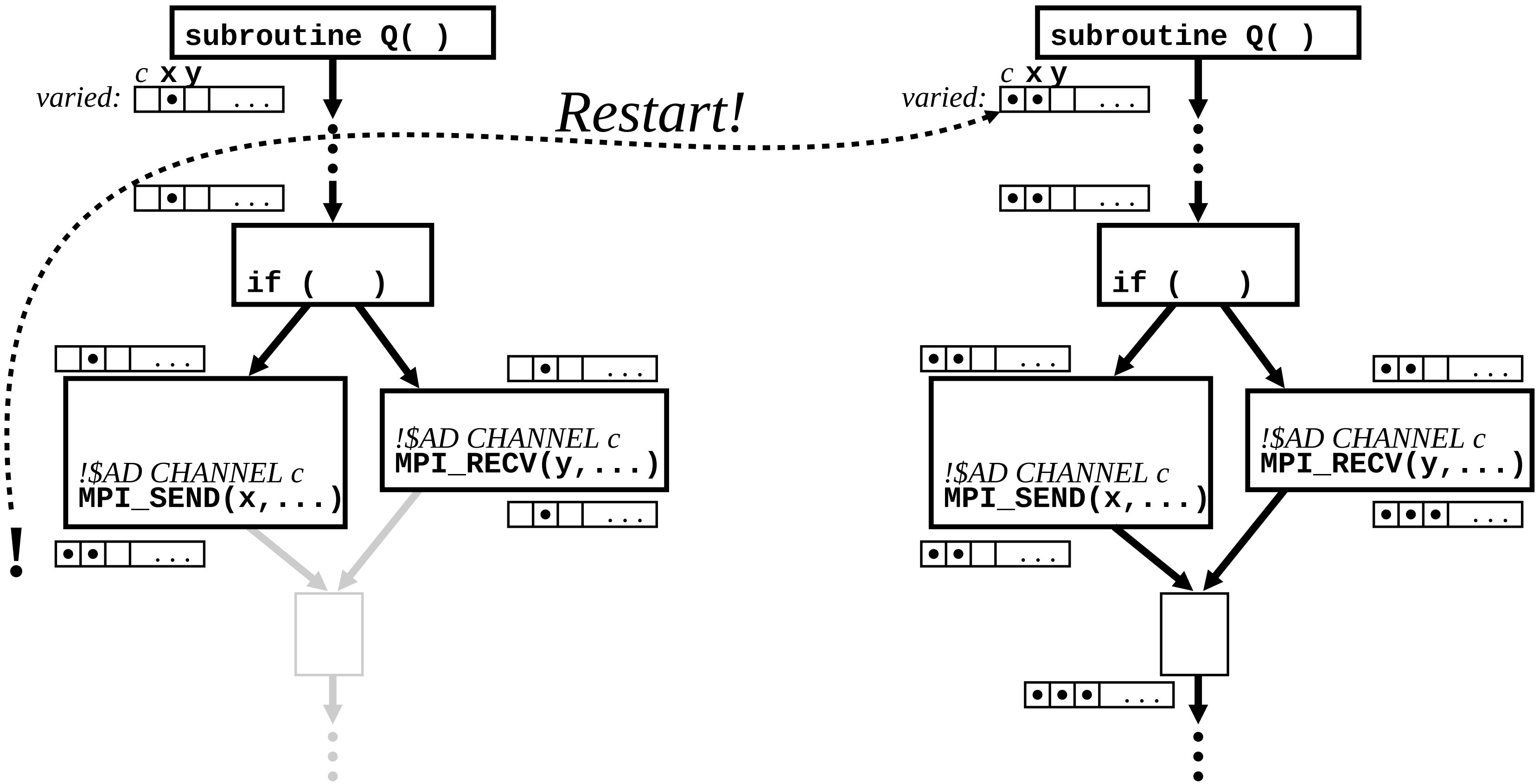
Channels and Flow graph local restart



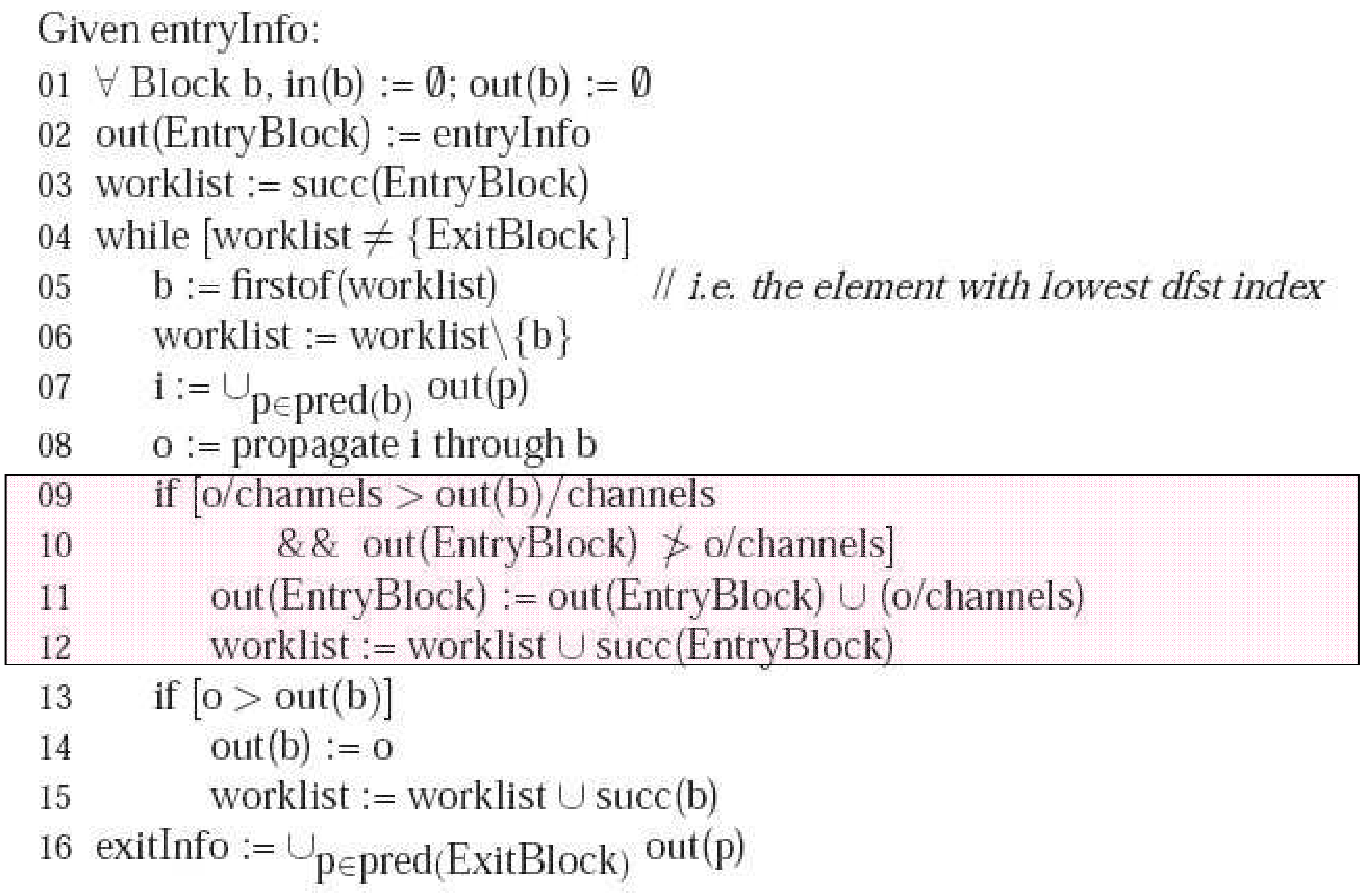
Channels and Flow graph local restart



Channels and Flow graph local restart



Simple algorithm extension



```
Given entryInfo:
$\forall$ Block b, in(b) := $\emptyset$; out(b) := $\emptyset$
out(EntryBlock) := entryInfo
worklist := succ(EntryBlock)
while [worklist $\neq$ {ExitBlock}]
    b := firstof(worklist)            // i.e. the element with lowest dfst index
    worklist := worklist \ {b}
    i := $\cup_{p \in pred(b)}$ out(p)
    o := propagate i through b
    if [o/channels > out(b)/channels
            && out(EntryBlock) $\not>$ o/channels]
        out(EntryBlock) := out(EntryBlock) $\cup$ (o/channels)
        worklist := worklist $\cup$ succ(EntryBlock)
    if [o > out(b)]
        out(b) := o
        worklist := worklist $\cup$ succ(b)
exitInfo := $\cup_{p \in pred(ExitBlock)}$ out(p)
```

Performance discussion

- Termination:

  the set of values for the data-flow info is finite

  during the analysis this info grows

  local restarts are in a finite number

  → the analysis terminates

- Execution time: depends on the number of channels
- Accuracy: related to the number of channels

  more channels → more accurate analysis but increased execution time

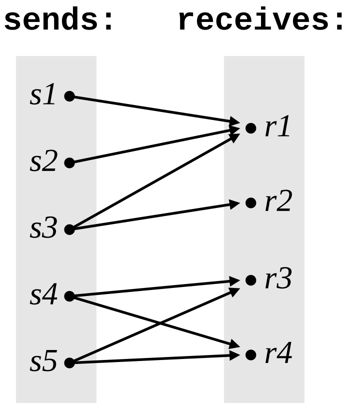
Choosing a good set of channels

Needs a test to match send's with receive's (using source, destination, tag, communicator or user's directives)

Then choose channels:

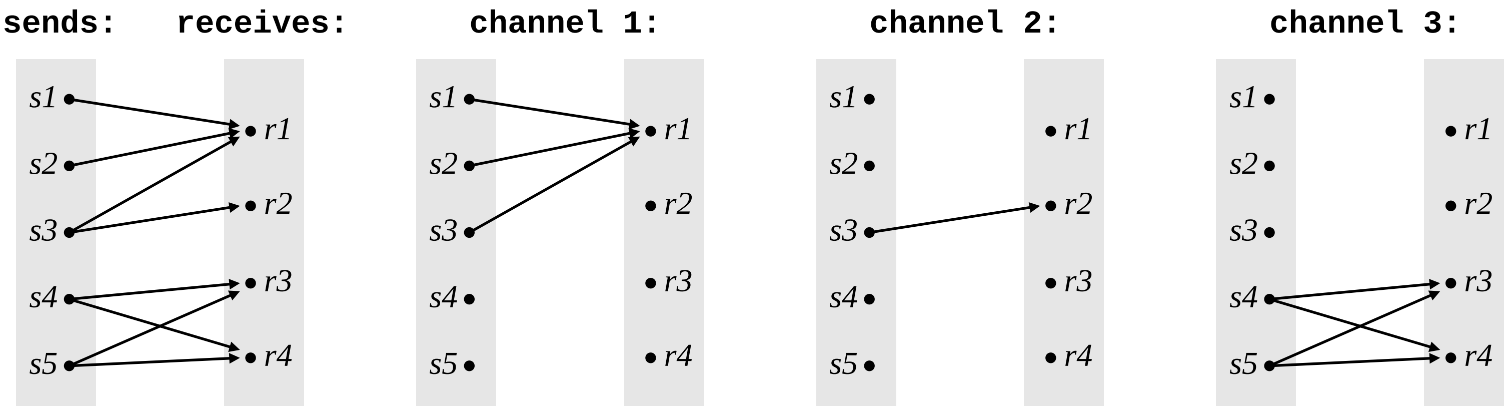
A good set of channels must not introduce artificial communication and must be as small as possible

⇒ minimal biclique edge cover



Choosing a good set of channels

- Minimal biclique edge cover of a communication bipartite graph with 3 channels:



Implementation

- Define properties of MPI procedures in a special file.
  Distinguish collective and point-to-point MPI calls.
  Define a channel only for point-to-point calls.

- Implement flow graph local restart for all data-flow analyses ⇒ inheritance

Validation on a CFD code

Aironum: Unsteady turbulent Navier-Stokes

100 000 lines

SPMD parallel with MPI

Tangent differentiation ok: 0.49s (original code: 0.38s)

Adjoint under development

Further work & difficulties

- Create new variables in differentiated MPI calls for tag status and error and propagate them

```
CALL MPI_SEND(xd, 1, mpi_real, 1, tagd, mpi_comm_world, coded)
CALL MPI_SEND(x,  1, mpi_real, 1, tag,  mpi_comm_world, code)
```

- Question: MPI_WAIT, MPI_WAITANY, MPI_WAITALL

  → Adjoinable MPI, wrap MPI calls?

  → Can a static analysis find all matches?

  Should it be dynamic?

Further work & difficulties

- Create new variables in differentiated MPI calls for tag status and error and propagate them

```
CALL MPI_SEND(xd, 1, mpi_real, 1, tagd, mpi_comm_world, coded)
CALL MPI_SEND(x,  1, mpi_real, 1, tag,  mpi_comm_world, code)
```

- Question: MPI_WAIT, MPI_WAITANY, MPI_WAITALL

  → Adjoinable MPI, wrap MPI calls?

  → Can a static analysis find all matches?
  Should it be dynamic?

Thank you for your attention