# PAGAI: a Path Sensitive Static Analyzer

Julien Henry David Monniaux Matthieu Moy

Verimag

September 14, 2012

# Summary

# PAGAI

Computes numerical invariants from an LLVM bitcode file

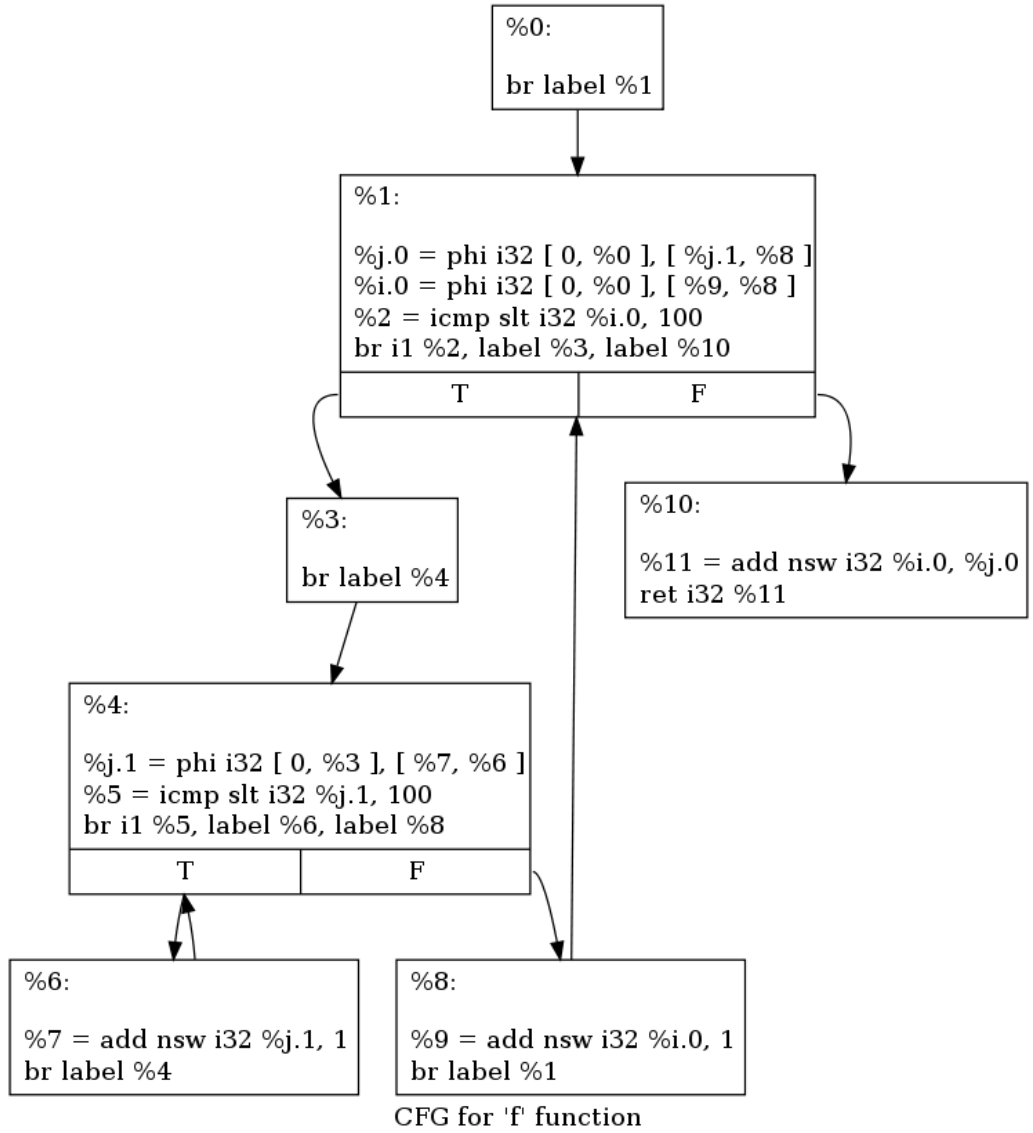
LLVM bitcode is generated by clang or llvm-gcc :

- clang supports C, C++, ...
- llvm-gcc supports C, C++, Fortran and Ada

# LLVM IR

```
int f() {
  int i = 0;
  int j = 0;

  while (i < 100) {
    j = 0;
    while (j < 100) {
      j++;
    }
    i++;
  }
  return i+j;
}
```



CFG for 'f' function

# Principle

Fully automatic discovering of invariants : Abstract Interpretation framework.

Computes an invariant per basic block :

- Keep as few dimensions as possible for the abstract value.
- Computes the image of an abstract value by a path/transition en bloc.

# Path Analysis

Several algorithm implemented in PAGAI compute an invariant only at a subset of the basic blocks (e.g loop headers)

- expand the rest of the control flow
- distinguish all the paths between two points
- avoid explicit enumeration of exponential set using SMT

# Example

For each loop header, Pagai returns an invariant:

```
int main() {
        int x = 0;
        int y = 0;
        while (1) {
                /* invariant:
                102 + -1 * x + -1 * y >= 0
                y >= 0
                x + -1 * y >= 0
                */
                if (x <= 50) y++;
                else y--;
                if (y < 0) break;
                x++;
        }
}
```

# Assert

```
int main() {
        int x = 0;
        int y = 0;
        while (1) {
                /* invariant:
                ...
                */
                if (x <= 50) y++;
                else y--;
                if (y < 0) break;
                x++;
        }
        /* assert OK */
        assert(x == 102);
}
```

# Assume

```
void rate_limiter() {
  int x_old;
  int x;
  x_old = 0;
  while (1) {
    /* invariant:
      100000 + -1 * x >= 0
      100000 + x >= 0
    */
    x = input();
    assume (x >= -100000 && x <= 100000);
    if (x > x_old+10) x = x_old+10;
    if (x < x_old-10) x = x_old-10;
    x_old = x;
  }
/* UNREACHABLE */}
```

# Undefined Behaviour

Using IOC: http://embed.cs.utah.edu/ioc/ : instrument generated code

```
int sum(unsigned n, int tab[n]) {
  if (n > 0) {
    int s = 0;
    for(unsigned i=0 /* invariant: ... */ ; i<n; i++)
      /* UNDEFINED BEHAVIOUR
      += : Signed Addition Overflow */
      s += tab[i];
    }
    return s;
  }
  return 0;
}
```

# Techniques Implemented

- Classic Abstract interpretation
- Guided static analysis (Gopan & Reps SAS'07)
  Ascending sequence of the subset of the transitions
- Path Focusing (Monniaux & Gonnord SAS'11)
  Enumerate paths using SMT
- Combined technique (Henry, Monniaux & Moy SAS'12)
  Ascending sequence of the subset of the paths, SMT
- Combined technique with disjunctive invariants (Henry, Monniaux & Moy SAS'12)

# Summary

# Dimensions of the Abstract Values

The CFG is in SSA form : **many variables**.

- Keep only Live variables

# Dimensions of the Abstract Values

The CFG is in SSA form : **many variables**.

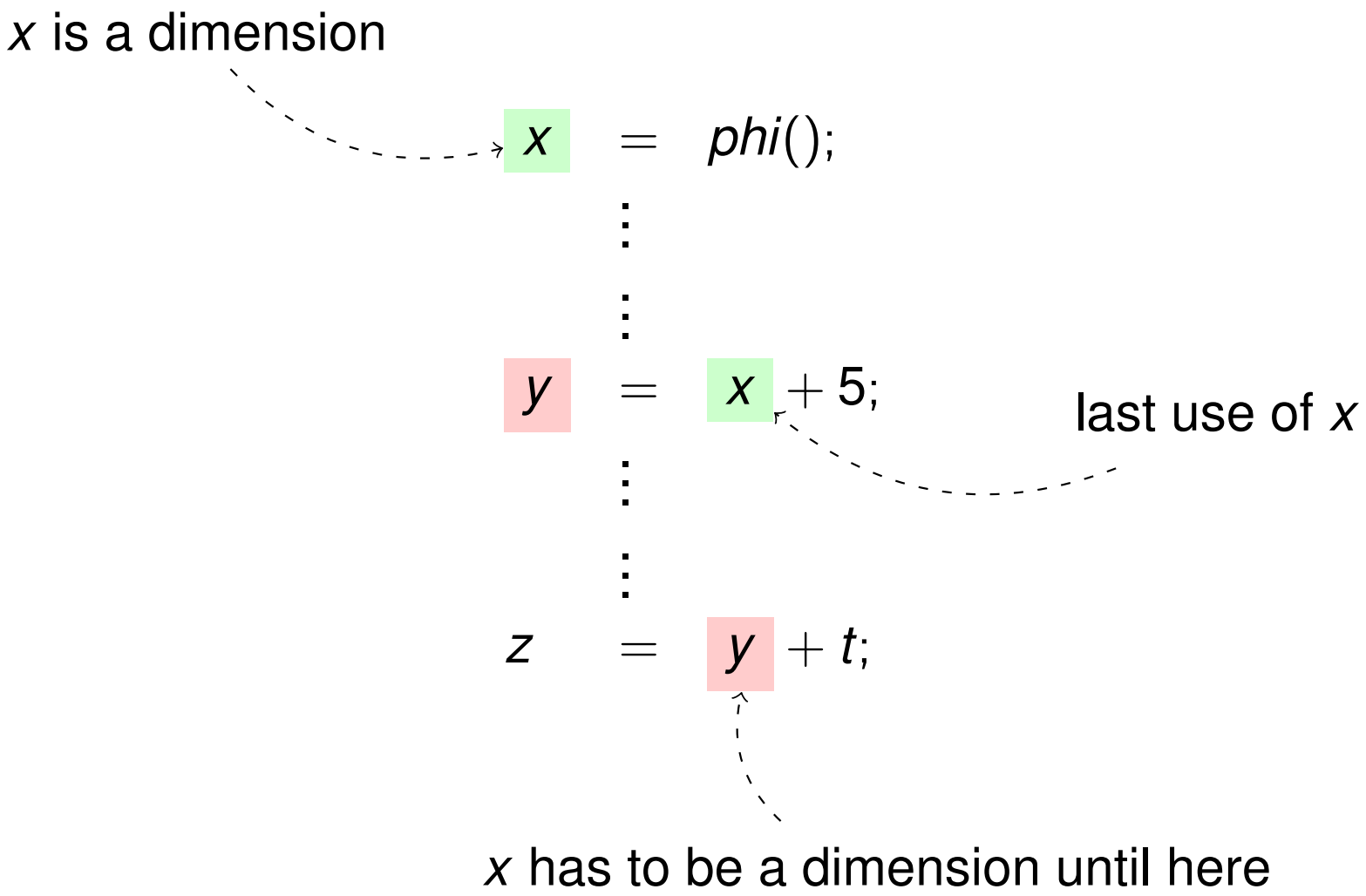
- Keep only Live variables
- Idea : Remove variables that are linear combination of others

# Dimensions of the Abstract Values

The CFG is in SSA form : **many variables**.

- Keep only Live variables
- Idea : Remove variables that are linear combination of others
  - Does not work. . .

# Dimensions of the Abstract Values



# Dimensions of the Abstract Values

## Definition

A variable $v$ is **live by linearity** at a control point $p$ if and only if one of these conditions holds:

- $v$ is live in $p$.
- There is a variable $v'$ , defined as a linear combination of other variables $v_1, v_2, \ldots, v_n$ , so that $\exists i \in 1, \ldots, n, v = v_i$, and $v'$ is live by linearity in $p$.

An LLVM value is a dimension in the abstract domain if and only if it is live by linearity and it is not defined as a linear combination of other LLVM values.

# Apron

B. Jeannet and A. Mine. CAV'2009.

Apron: Numerical abstract domain library.

- BOX intervals library
- OCT octagon library
- NEWPOLKA Convex Polyhedra and Linear Equalities library

Also provides an interface to the Parma Polyhedra Library

# Parallel Assignment

A BasicBlock (or a path) amounts to a parallel assignment operation between live-by-linearity variables

$$(v_1, \ldots, v_n) \mapsto (f_1(v_1, \ldots, v_n), \ldots, f_n(v_1, \ldots, v_n))$$

Benefits:

- limits the number of dimension in the abstract values
- large parallel assignment for path focusing techniques
- more precise (ex: $y = x; z = x - y;$)

# SMT Solvers

Pagai uses SMT solvers for Path-focusing techniques

- SMT-Lib2 interface through a pipe (Z3, Mathsat5, SMTInterpol)
- Yices API
- Microsoft Z3 API

# Limitations

- no memory model
  - LOAD $x \Rightarrow x \in (-\infty, +\infty)$
  - STORE $x$ does nothing

  Remark : We use an LLVM pass (mem2reg) that lifts most memory accesses to scalar variables.
- intraprocedural
  - a function call has the same effect as a LOAD
  - we inline...
- Integer are mathematical integers ($\mathbb{Z}$)
- Floating point variables are reals (in $\mathbb{R}$ or $\mathbb{Q}$)

# Summary

# Motivation

Hard to know which technique performs better.

We have to experiment and compare.

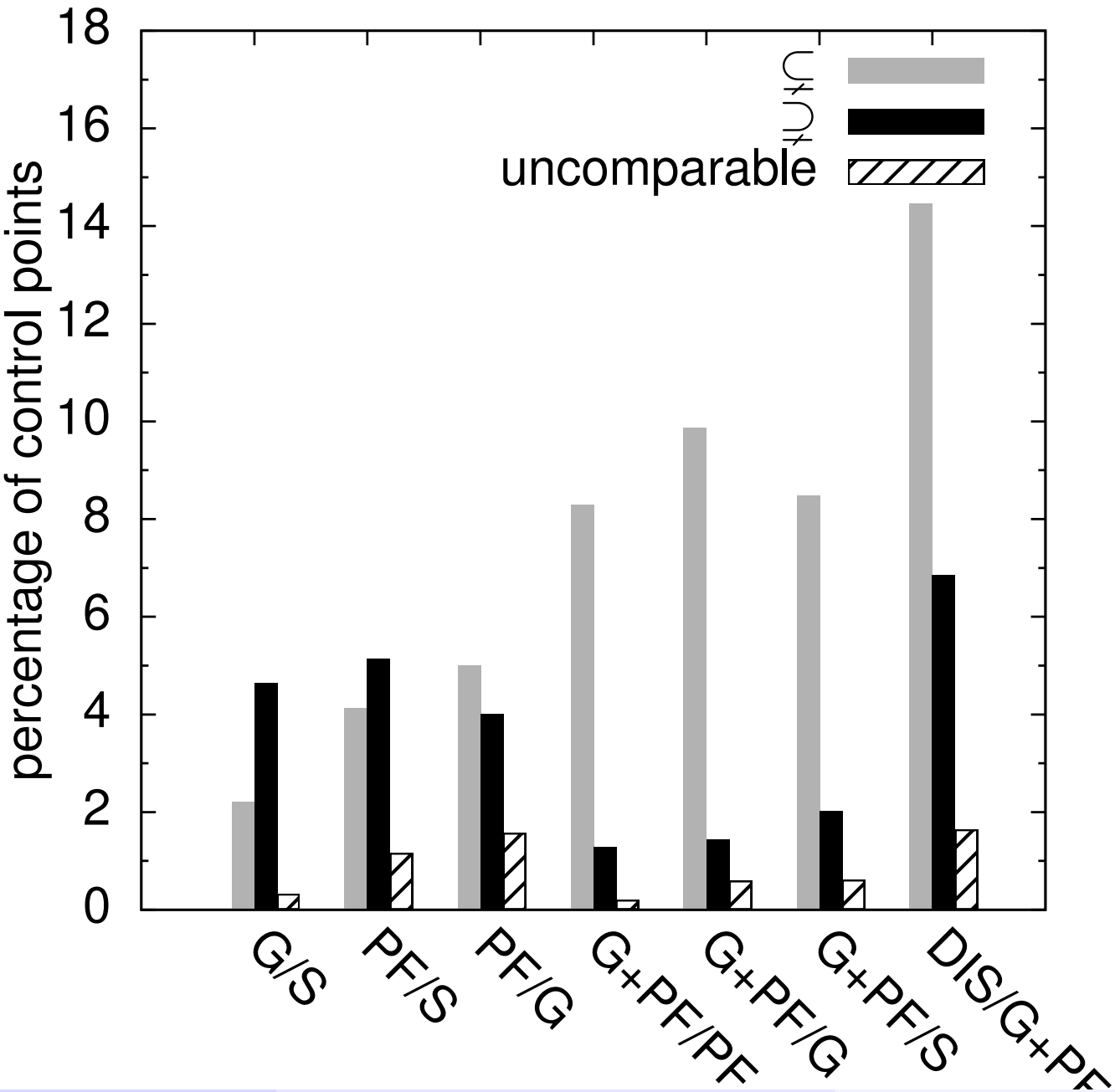
# Experiments

Various experiments have been done :

- benchmarks for WCET analysis or termination
- various GNU programs
  - grep, gnuchess, gnugo, gzip, etc.

# Scalability

| Name | kLOC | $\lvert loops \rvert$ | Time (in seconds) | | | | |
|---|---|---|---|---|---|---|---|
| | | | S | G | PF | C | DIS |
| a2ps | 55 | 2012 | 23 | 74 | 34 | 115 | 162 |
| gawk | 59 | 902 | 15 | 46 | 12 | 40 | 50 |
| gnuchess | 38 | 1222 | 50 | 220 | 81 | 312 | 351 |
| gnugo | 83 | 2801 | 77 | 159 | 92 | 766 | 1493 |
| grep | 35 | 820 | 41 | 85 | 22 | 65 | 122 |
| gzip | 27 | 494 | 22 | 268 | 91 | 303 | 230 |
| lapack | 954 | 16422 | 294 | 3740 | 3773 | 8159 | 10351 |
| make | 34 | 993 | 67 | 108 | 53 | 109 | 257 |
| tar | 73 | 1712 | 37 | 218 | 115 | 253 | 396 |


percentage of control points
0
2
4
6
8
10
12
14
16
18
G/S
PF/S
PF/G
G+PF/PF
G+PF/G
G+PF/S
DIS/G+PF
uncomparable

# Summary

# Conclusion

A new prototype of static analyzer for comparing techniques:

- Easy choice of abstract domain and technique
- A technique is implemented as an LLVM optimisation pass
- Design choices for performant path focusing techniques

# Future Work / Work in Progress

- Iterative analysis
- Correct handling of machine integers and floating points
- Pointers, Arrays

# How to Get Pagai

Pagai is an open-source software.

Freely downloadable here :

`http://forge.imag.fr/projects/pagai`