Machine Learning in Compilers

Program Analyses and Optimizations - DCC888

Fernando Magno Quintão Pereira

fernando@dcc.ufmg.br

What is Machine Learning?

What is Machine Learning?

- Program Synthesis

What is Machine Learning?

- ~~Program~~ Function Synthesis

What is Machine Learning?

- ~~Program~~ Function Synthesis - From Examples

What is Machine Learning?

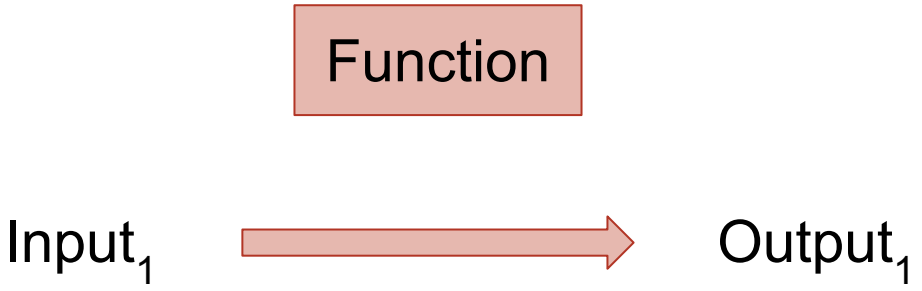
- ~~Program~~ Function Synthesis - From Examples

Function

???

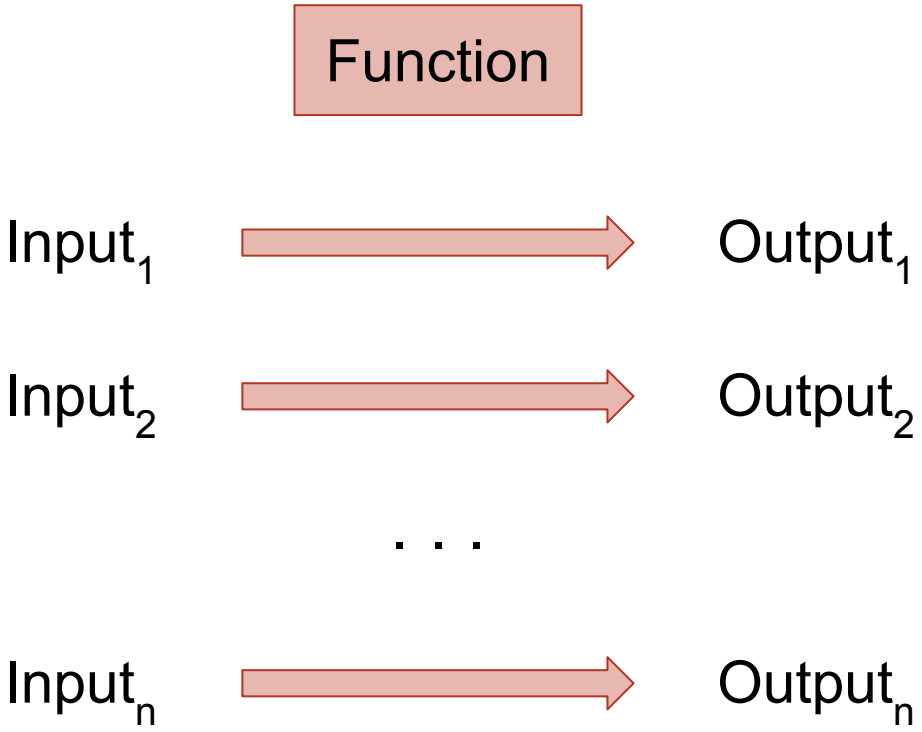
What is Machine Learning?

- ~~Program~~ Function Synthesis - From Examples



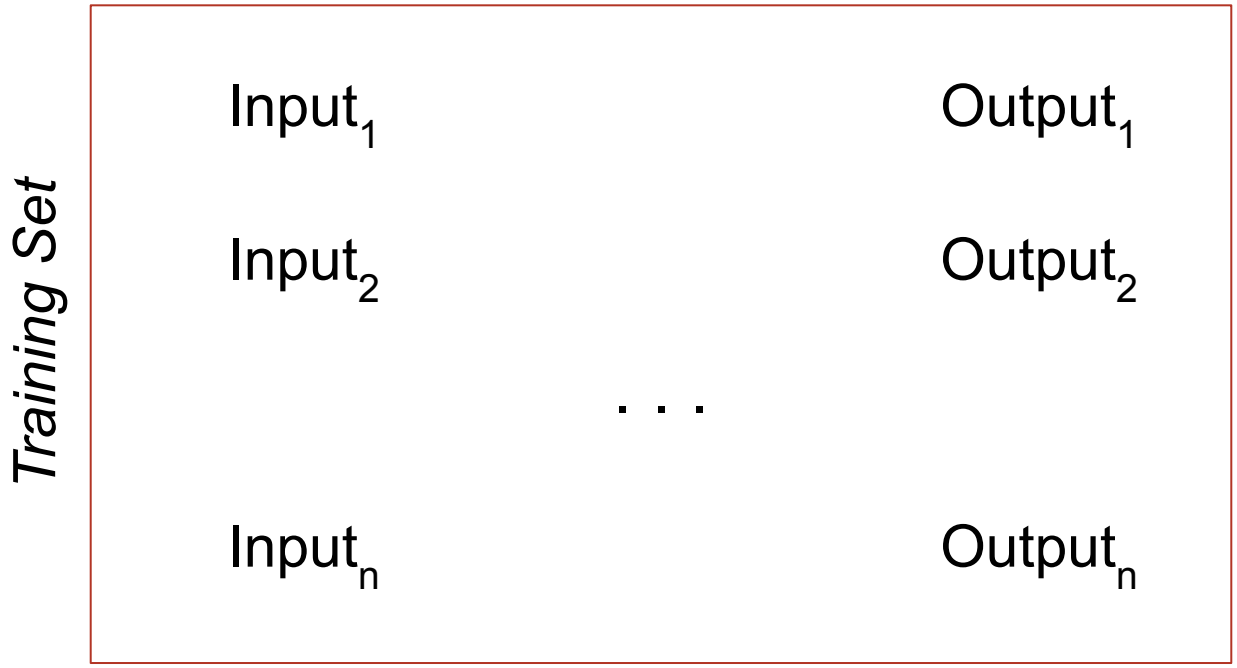
What is Machine Learning?

- ~~Program~~ Function Synthesis - From Examples



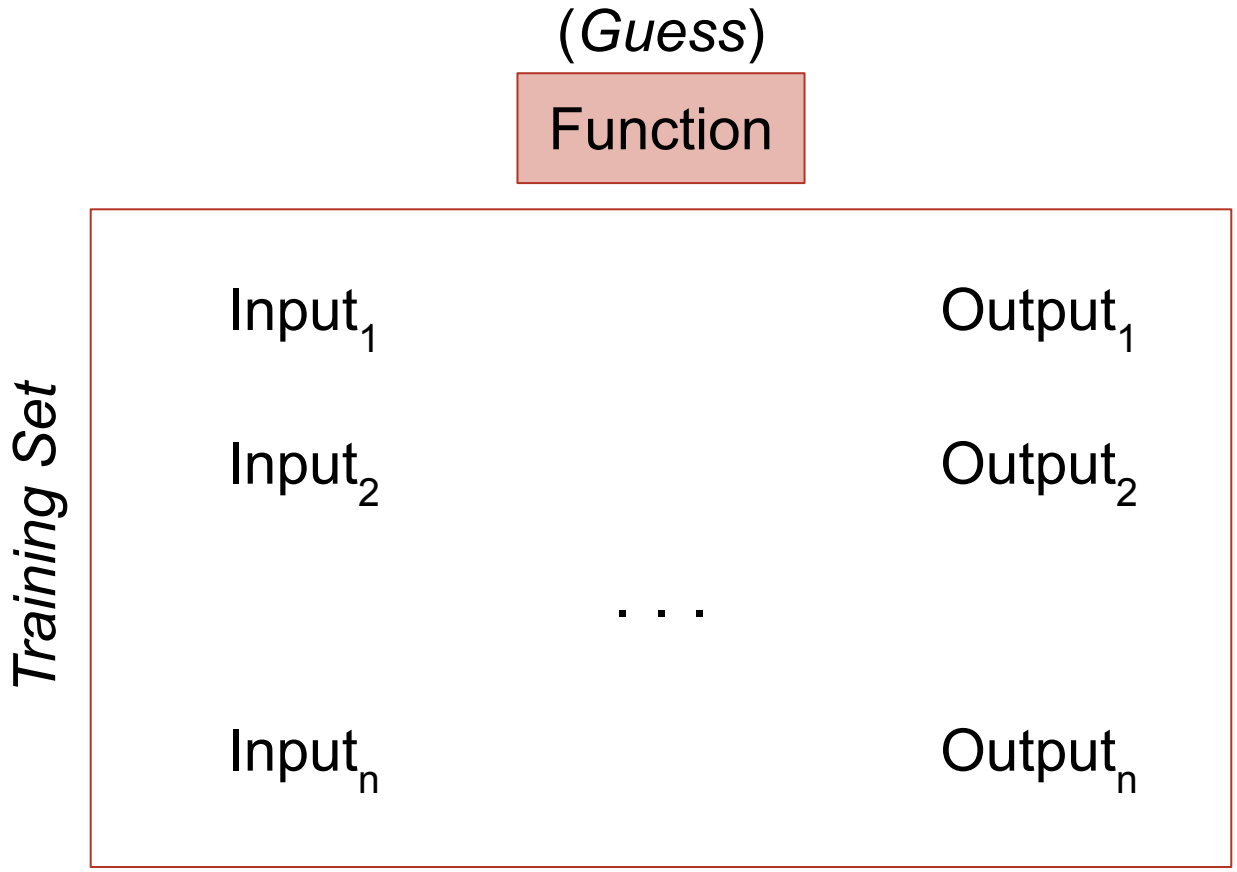
What is Machine Learning?

- ~~Program~~ Function Synthesis - From Examples



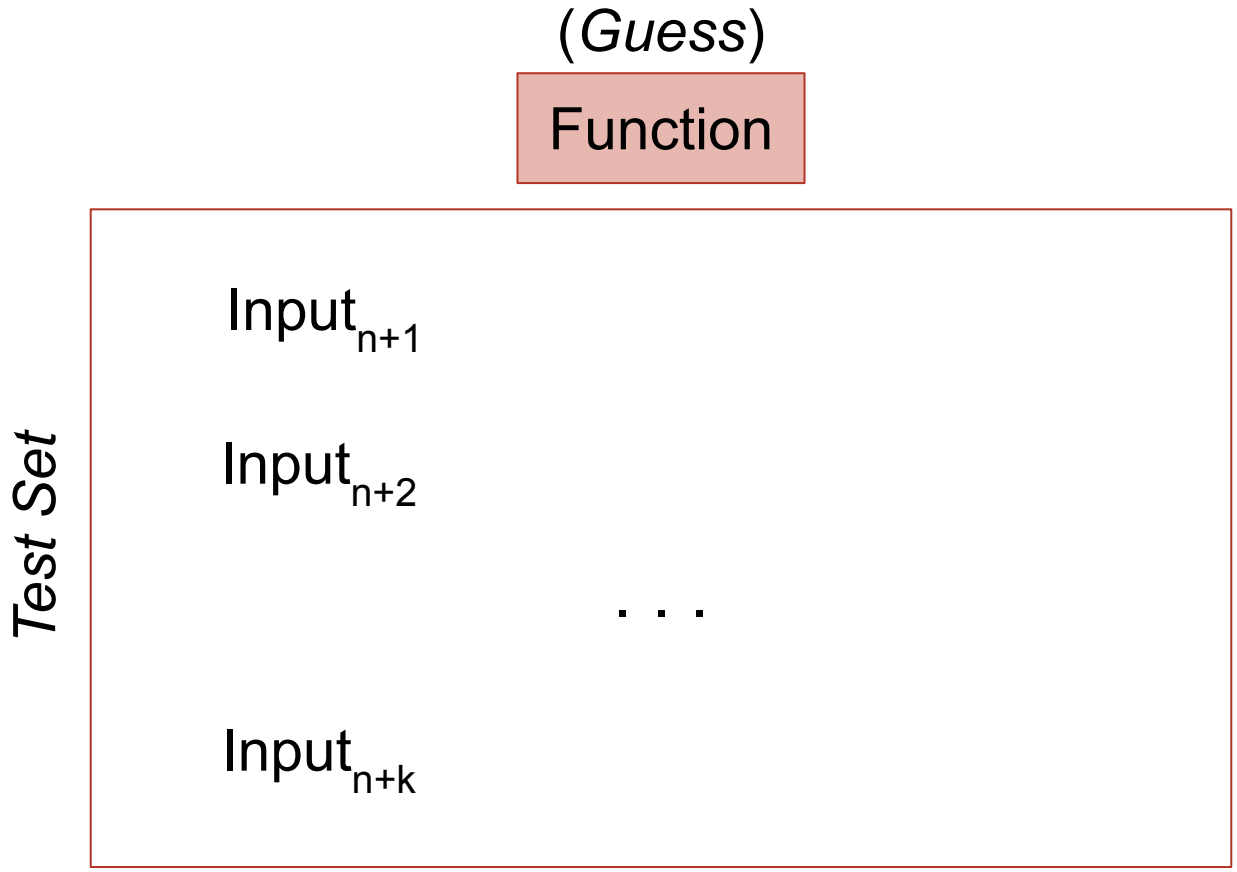
What is Machine Learning?

- ~~Program~~ Function Synthesis - From Examples



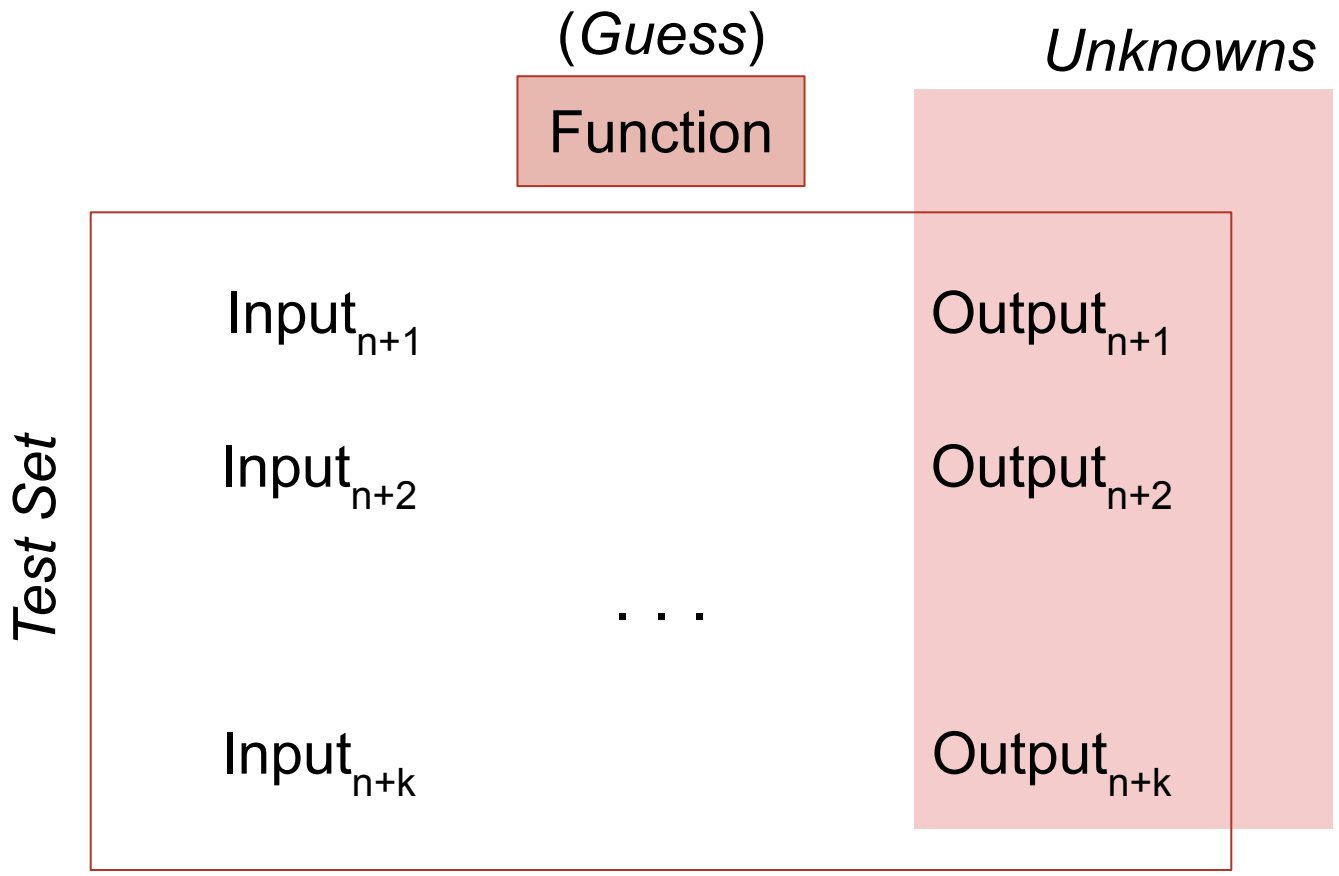
What is Machine Learning?

- ~~Program~~ Function Synthesis - From Examples



What is Machine Learning?

- ~~Program~~ Function Synthesis - From Examples



Why is this Process Stochastic?

Why is this Process Stochastic?

- Approximations are acceptable

Why is this Process Stochastic?

- Approximations are acceptable
- The exact function might not exist

Why is this Process Stochastic?

- Approximations are acceptable
- The exact function might not exist

Why is this Process Stochastic?

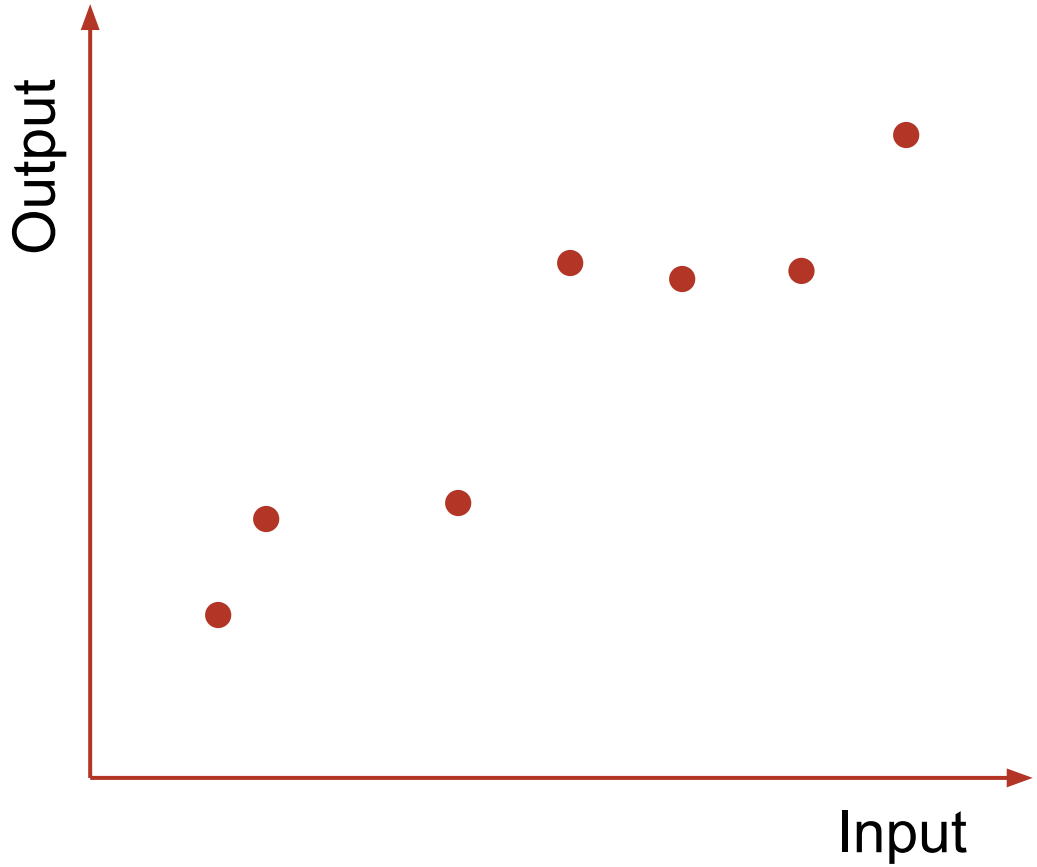
- Approximations are acceptable
- The exact function might not exist

Why is this Process Stochastic?

- Approximations are acceptable
- The exact function might not exist

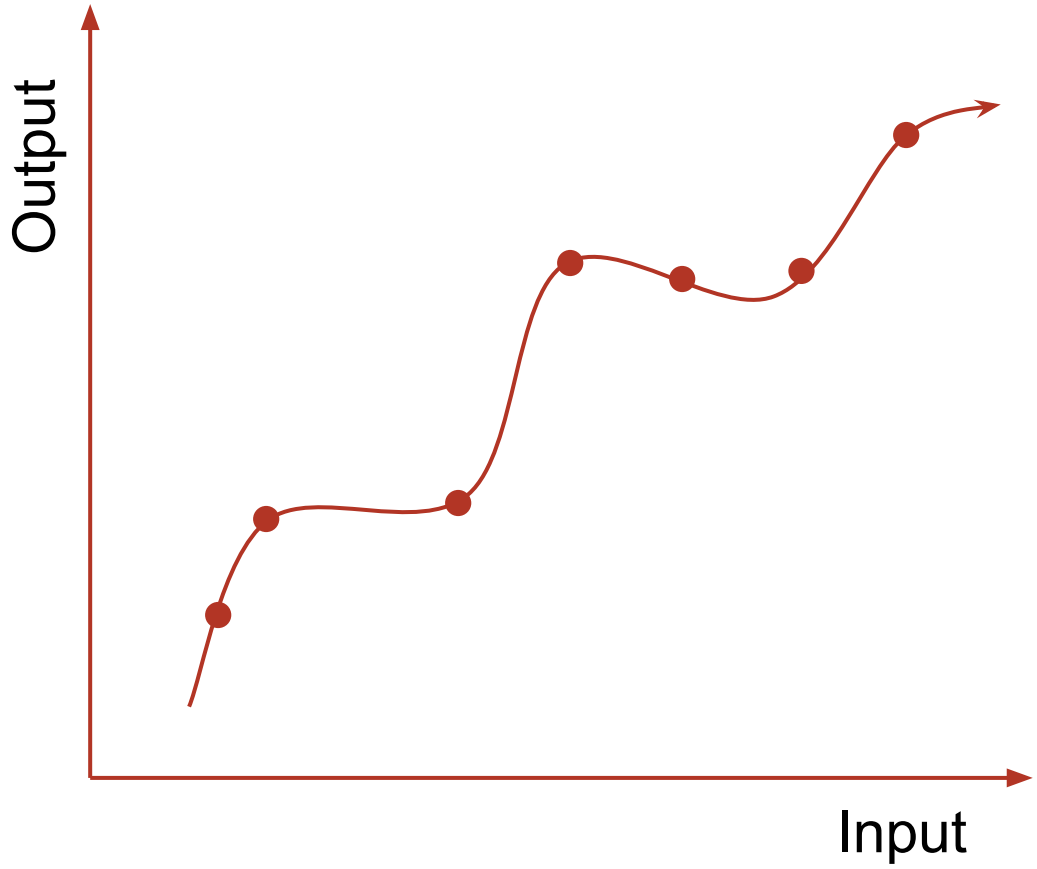
Why is this Process Stochastic?

- Approximations are acceptable



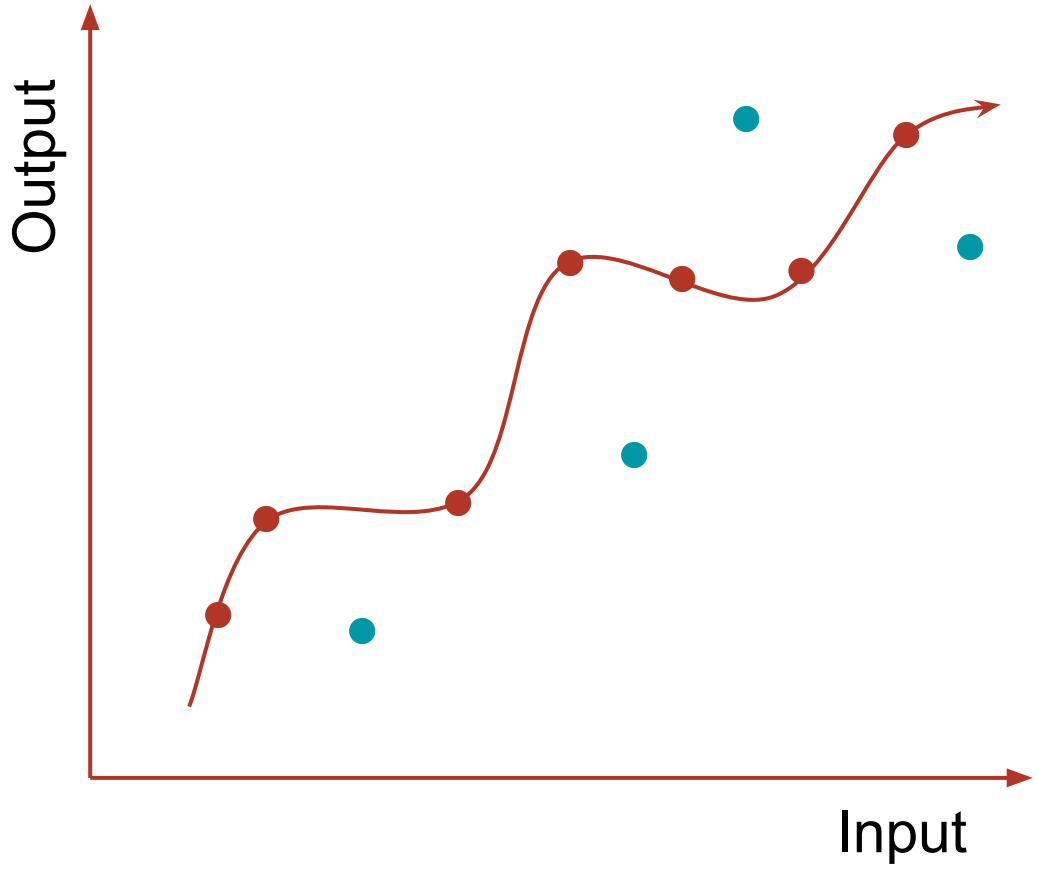
Why is this Process Stochastic?

- Approximations are acceptable



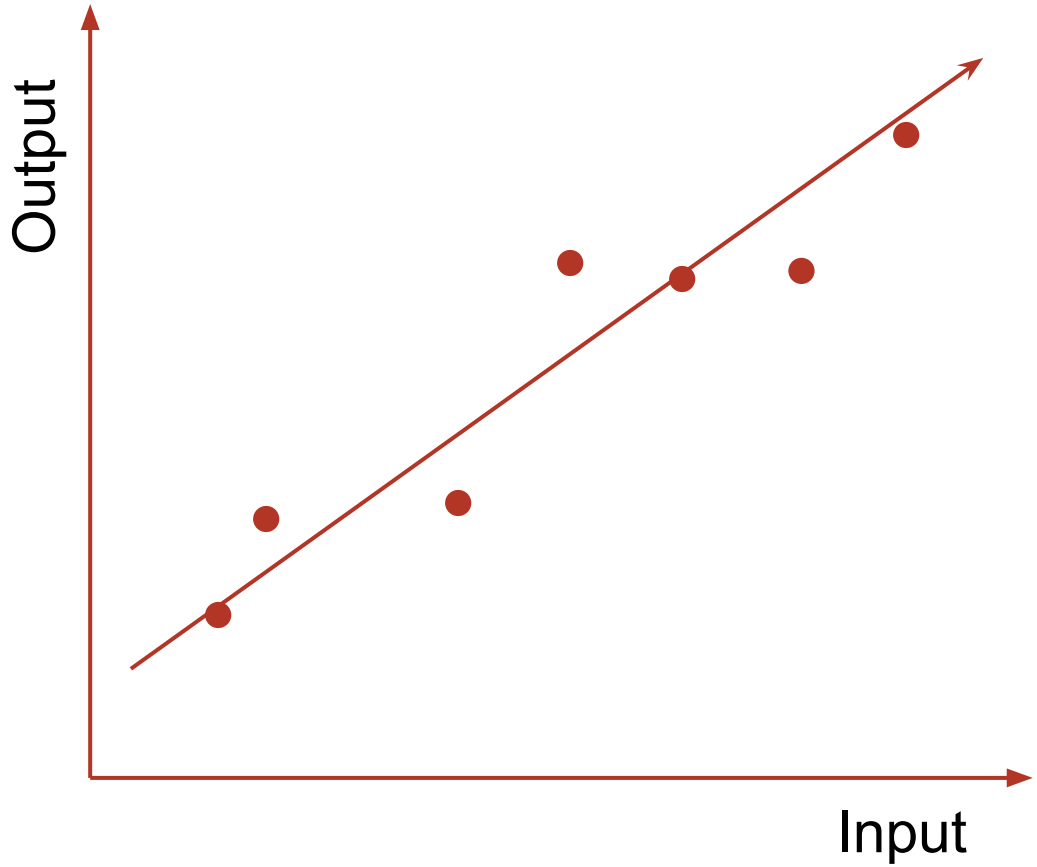
Why is this Process Stochastic?

- Approximations are acceptable



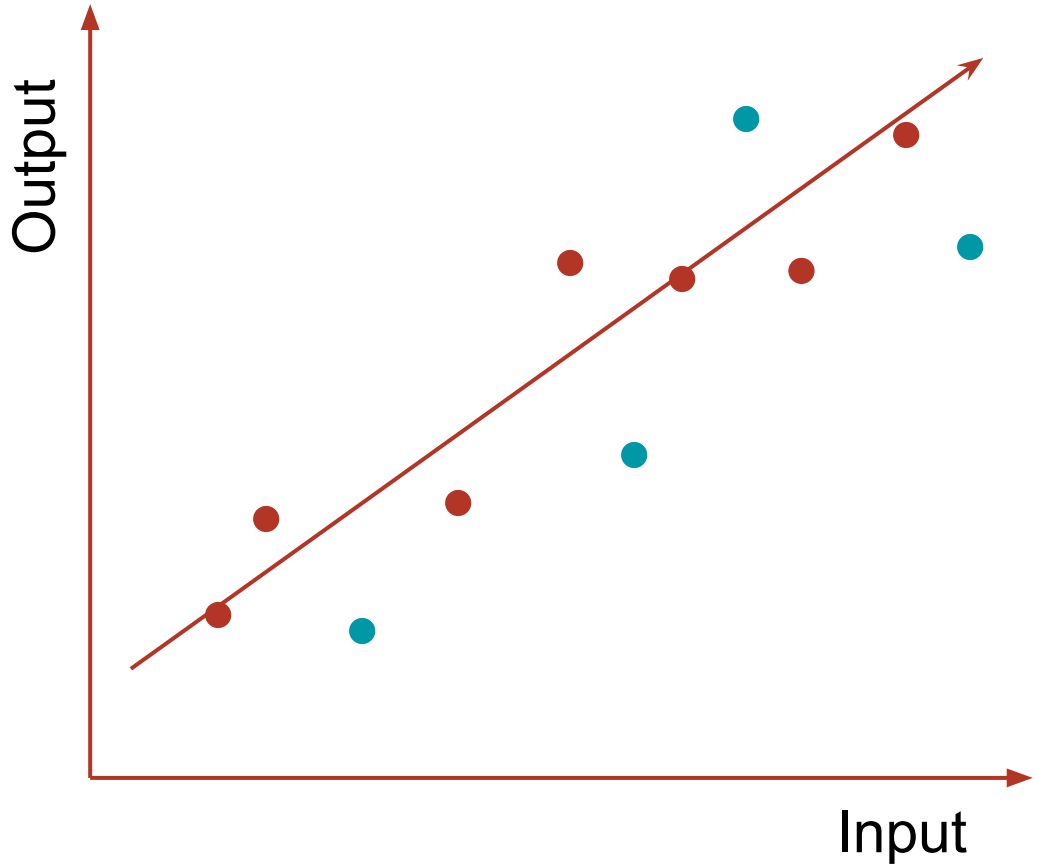
Why is this Process Stochastic?

- Approximations are acceptable



Why is this Process Stochastic?

- Approximations are acceptable



How are the Functions?

How are the Functions?

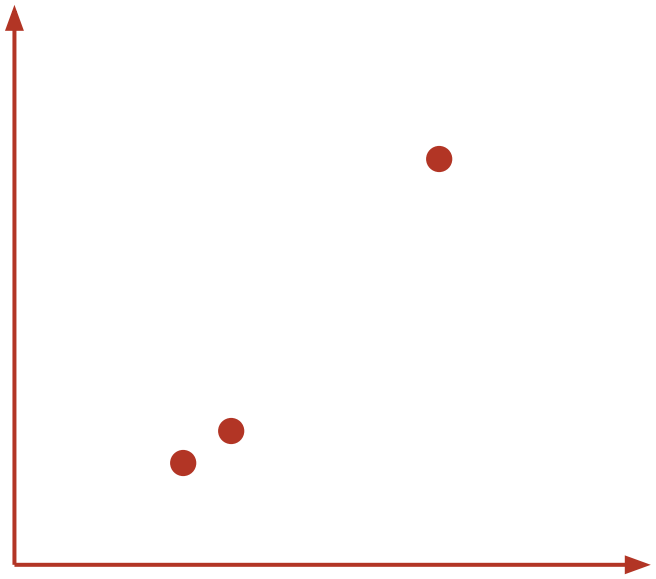
Regression

Classification

How are the Functions?

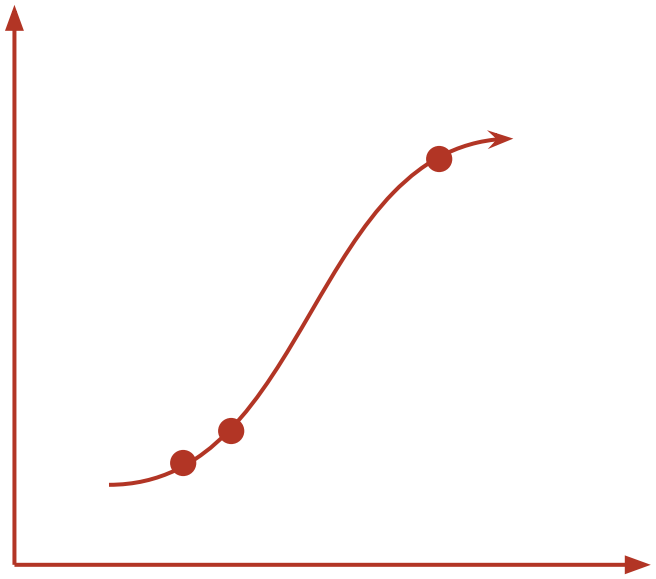
Regression

Classification



How are the Functions?

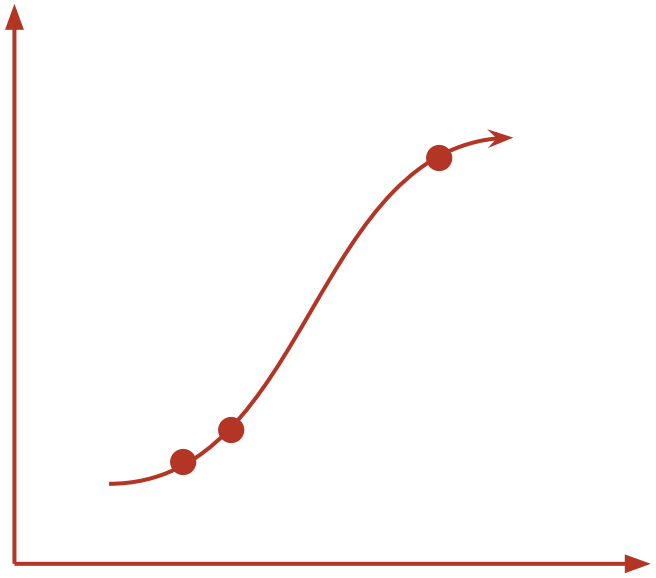
Regression



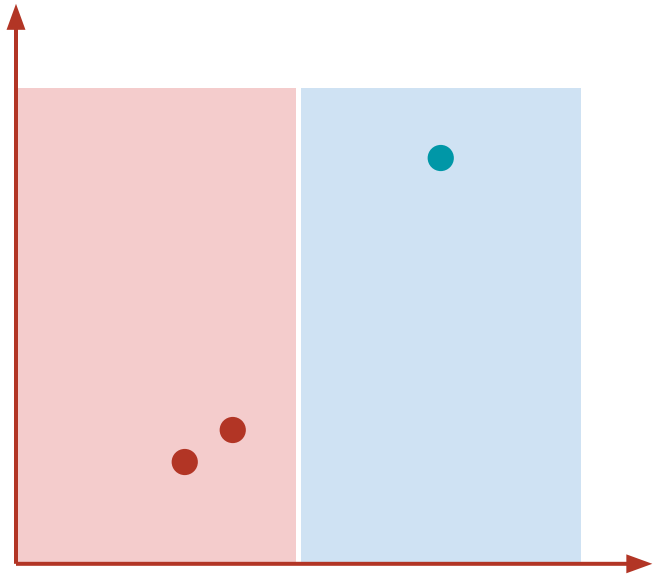
Classification

How are the Functions?

Regression



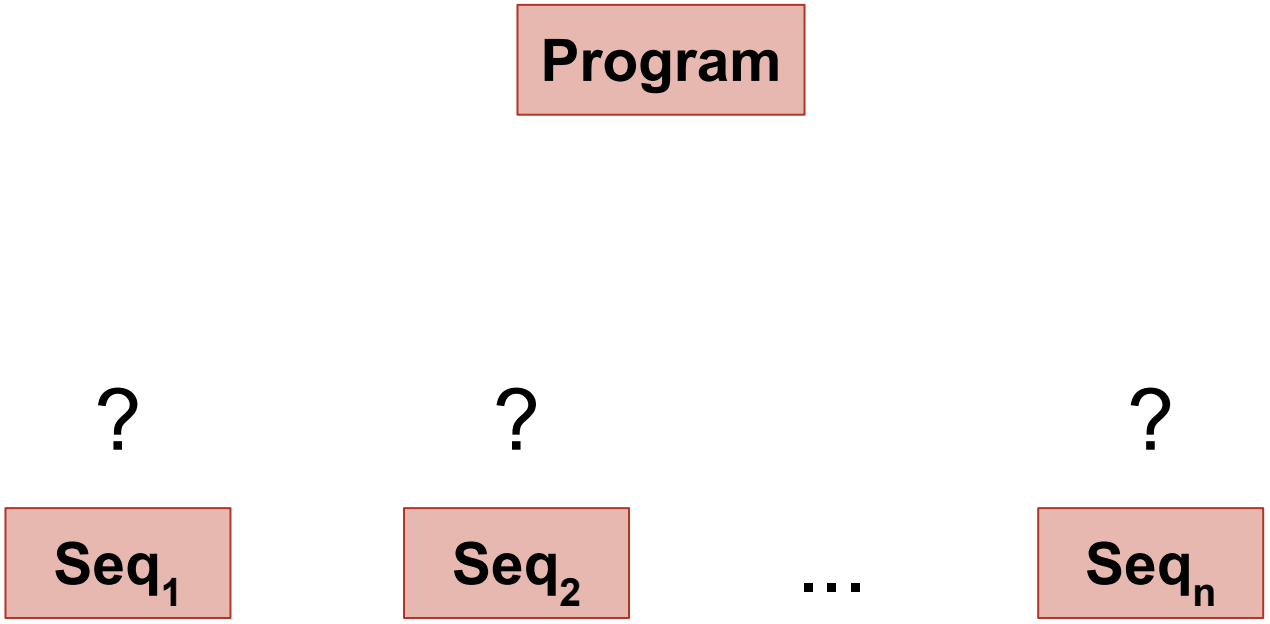
Classification



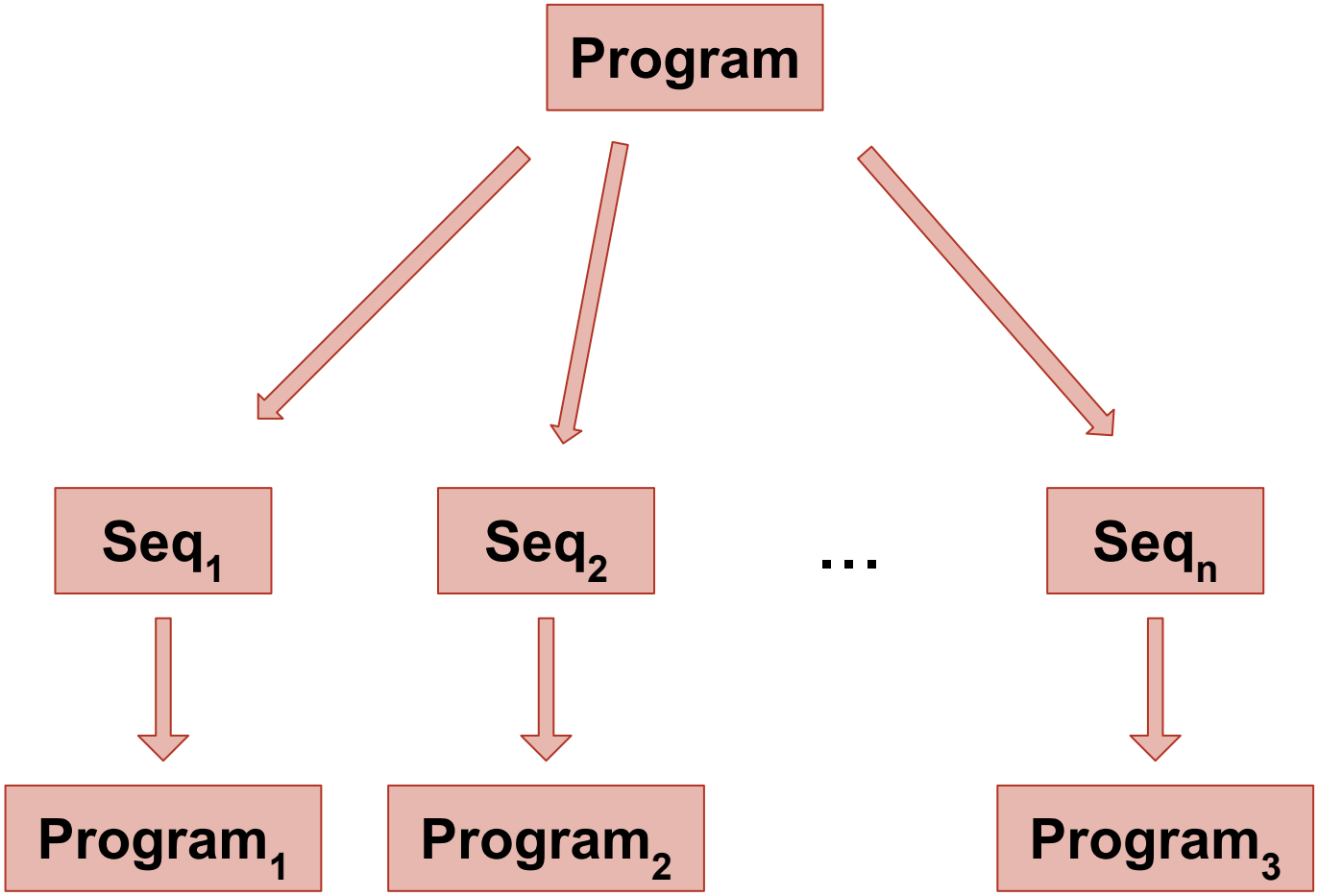
What to do with Compilers?

Example: Best Sequence of Optimizations

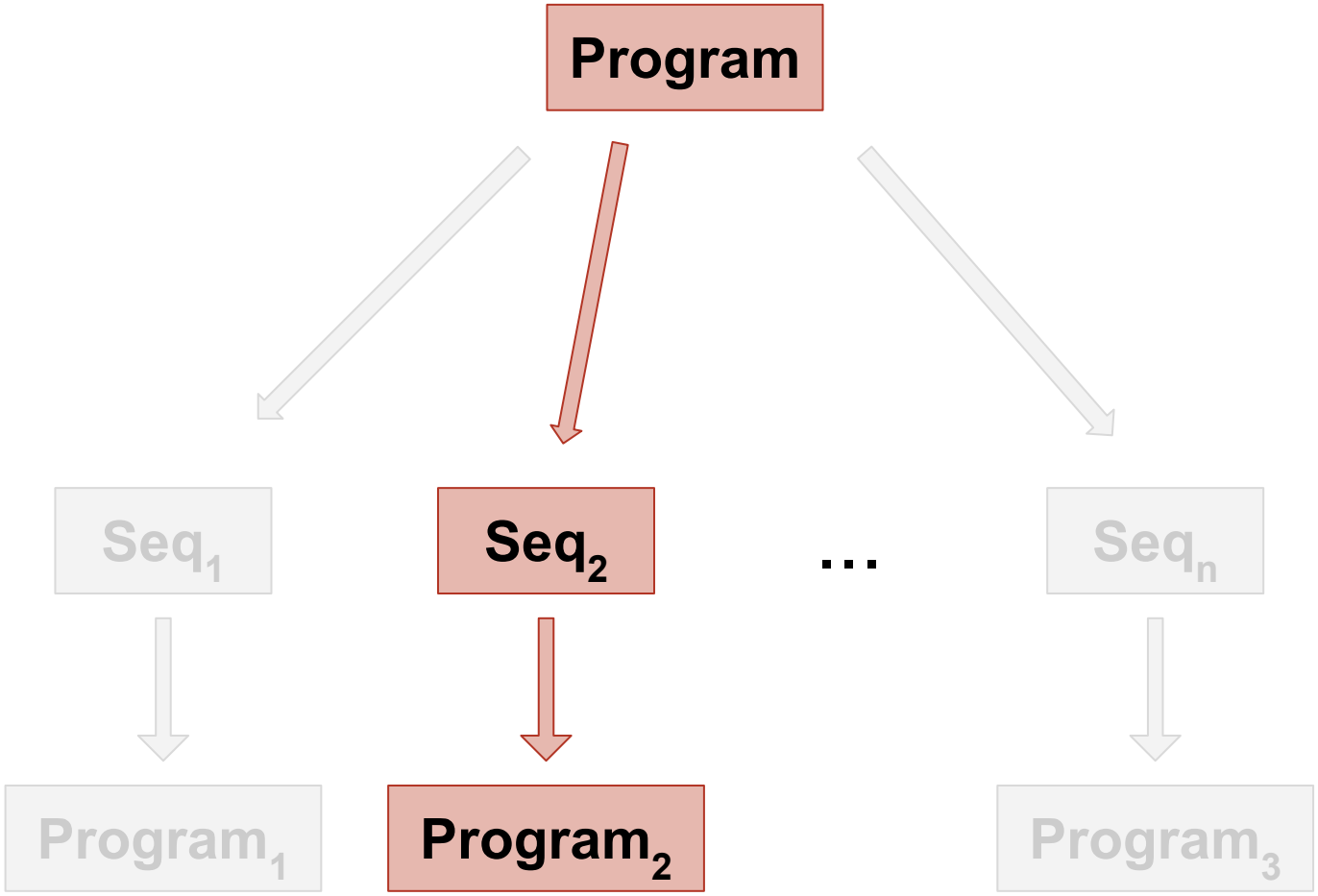
Example: Best Sequence of Optimizations



Example: Best Sequence of Optimizations



Example: Best Sequence of Optimizations



Example: Best Sequence of Optimizations

- Speed

John Cavazos, Michael F. P. O'Boyle:
Method-specific dynamic compilation using logistic regression.
OOPSLA 2006: 229-240

Example: Best Sequence of Optimizations

- Size

Anderson Faustino da Silva, Bernardo N. B. de Lima, Fernando Magno Quintão Pereira: *Exploring the space of optimization sequences for code-size reduction: insights and tools.* CC 2021: 47-58

Example: Best Sequence of Optimizations

- Size

Seq_1 Seq_2 ... Seq_n

How would you get candidate sequences?

Anderson Faustino da Silva, Bernardo N. B. de Lima, Fernando Magno Quintão Pereira: *Exploring the space of optimization sequences for code-size reduction: insights and tools.* CC 2021: 47-58

Example: Best Sequence of Optimizations

Seq_5

Seq_4

Seq_3

Seq_2

Seq_1

How would you get candidate sequences?

Anderson Faustino da Silva, Bernardo N. B. de Lima, Fernando Magno Quintão Pereira: *Exploring the space of optimization sequences for code-size reduction: insights and tools.* CC 2021: 47-58

Example: Best Sequence of Optimizations

Seq_5	-loop-rotate -sroa -correlated-propagation -indvars -gvn -tailcallelim -instcombine -jump-threading -reassociate -simplifycfg -instcombine -early-cse-memssa
Seq_4	-loop-vectorize -sroa -gvn -instcombine -simplifycfg -instcombine -licm -gvn -correlated-propagation -jump-threading -mldst-motion -early-cse-memssa -instcombine -simplifycfg -instsimplify
Seq_3	-mem2reg -early-cse-memssa -correlated-propagation -instcombine -reassociate -simplifycfg -early-cse-memssa -instcombine -licm -jump-threading -simplifycfg -dse -reassociate -early-cse-memssa -instcombine
Seq_2	-sroa -early-cse-memssa -reassociate -instcombine -simplifycfg -licm -speculative-execution -jump-threading -early-cse-memssa -simplifycfg -instcombine -simplifycfg
Seq_1	-mem2reg -jump-threading -instcombine -early-cse-memssa -jump-threading -licm -early-cse-memssa -sroa -simplifycfg -reassociate -instcombine -slp-vectorizer -early-cse-memssa

How would you get candidate sequences?

Anderson Faustino da Silva, Bernardo N. B. de Lima, Fernando Magno Quintão Pereira: *Exploring the space of optimization sequences for code-size reduction: insights and tools.* CC 2021: 47-58

Example: Best Sequence of Optimizations

-mem2reg -jump-threading -instcombine -early-cse-memssa -jump-threading -licm -early-cse-memssa -sroa -simplifycfg -reassociate -instcombine -slp-vectorizer -early-cse-memssa

How would you get candidate sequences?

Anderson Faustino da Silva, Bernardo N. B. de Lima, Fernando Magno Quintão Pereira: *Exploring the space of optimization sequences for code-size reduction: insights and tools.* CC 2021: 47-58

Example: Best Sequence of Optimizations

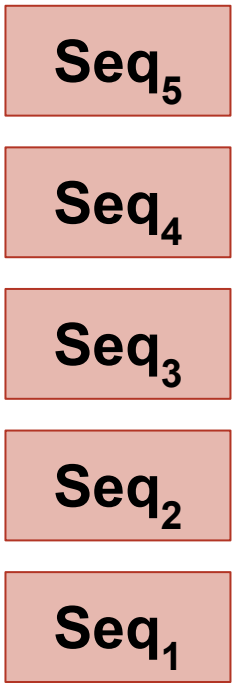
`https://llvm.org/docs/Passes.html`

-mem2reg -jump-threading -instcombine -early-cse-memssa -jump-threading -licm -early-cse-memssa -sroa -simplifycfg -reassociate -instcombine -slp-vectorizer -early-cse-memssa

How would you get candidate sequences?

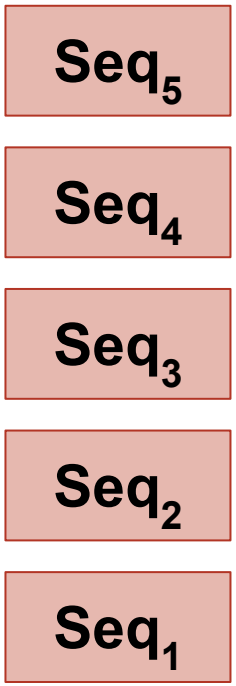
Anderson Faustino da Silva, Bernardo N. B. de Lima, Fernando Magno Quintão Pereira: *Exploring the space of optimization sequences for code-size reduction: insights and tools.* CC 2021: 47-58

Example: Best Sequence of Optimizations

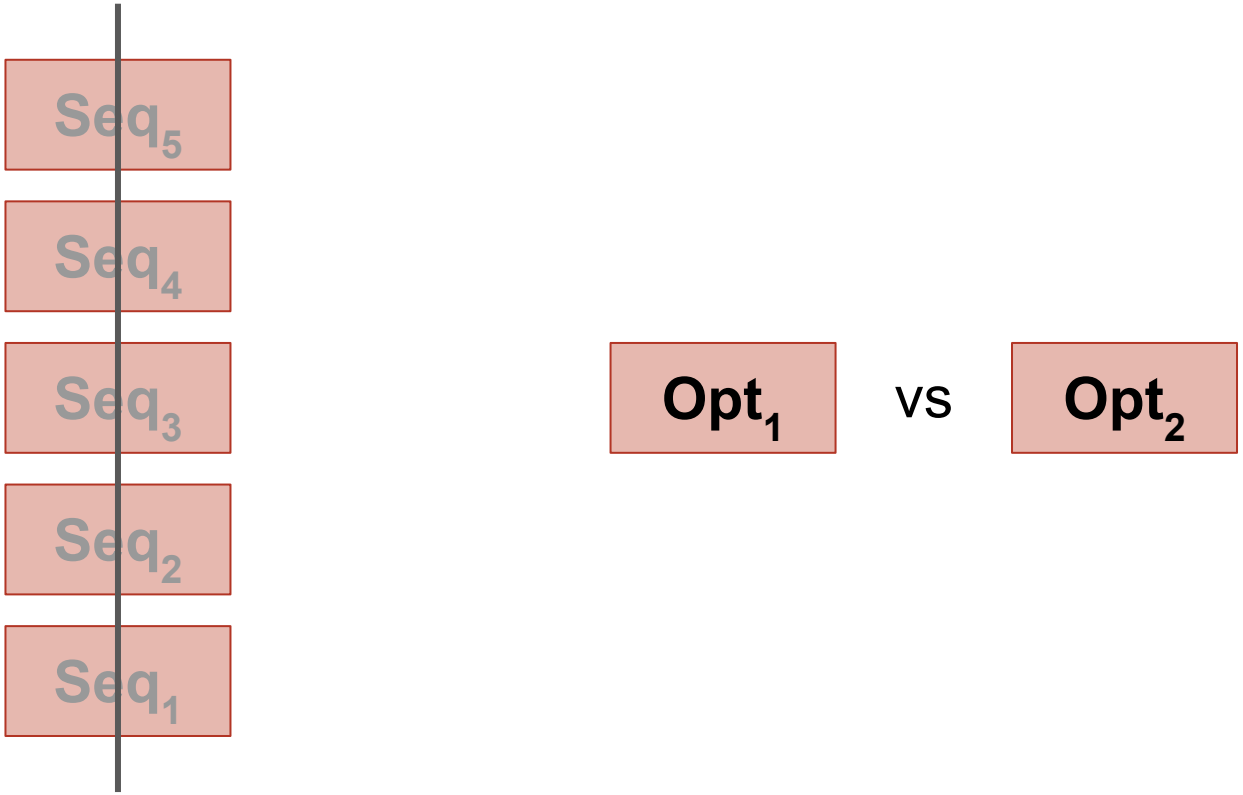


Anderson Faustino da Silva, Bernardo N. B. de Lima, Fernando Magno Quintão Pereira: *Exploring the space of optimization sequences for code-size reduction: insights and tools.* CC 2021: 47-58

Example: Best Optimization



Example: Best Optimization



Example: Best Optimization

Opt_1 vs **Opt_2**

John Cavazos, J. Eliot B. Moss, Michael F. P. O'Boyle:
Hybrid Optimizations: Which Optimization Algorithm to Use?
CC 2006: 124-138

Example: Best Optimization (Register Allocation)

Linear Scan vs **Graph Coloring**

John Cavazos, J. Eliot B. Moss, Michael F. P. O'Boyle:
Hybrid Optimizations: Which Optimization Algorithm to Use?
CC 2006: 124-138

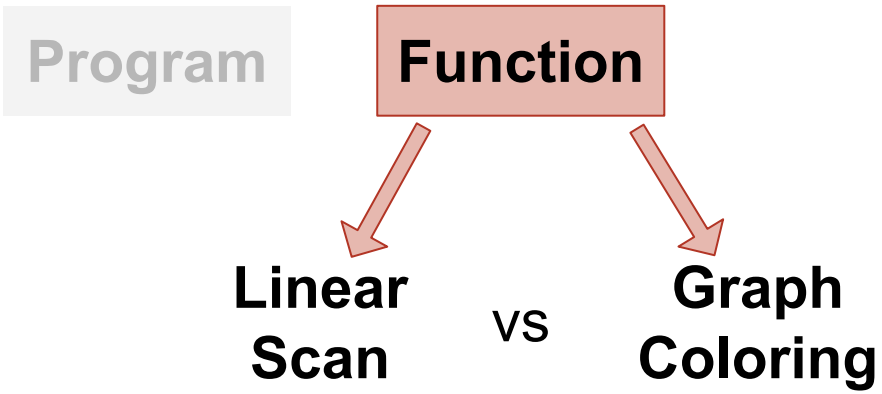
Example: Best Optimization (Register Allocation)

Program

Linear Scan vs **Graph Coloring**

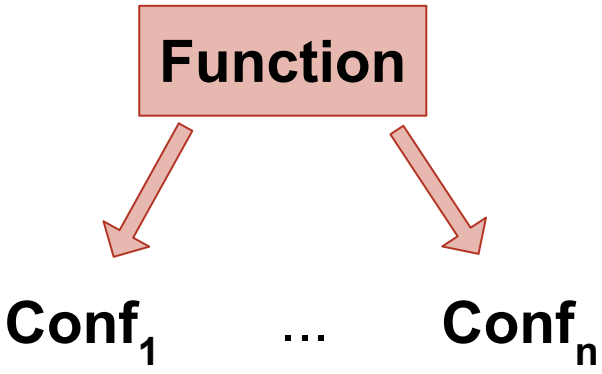
John Cavazos, J. Eliot B. Moss, Michael F. P. O'Boyle:
Hybrid Optimizations: Which Optimization Algorithm to Use?
CC 2006: 124-138

Example: Best Optimization (Register Allocation)

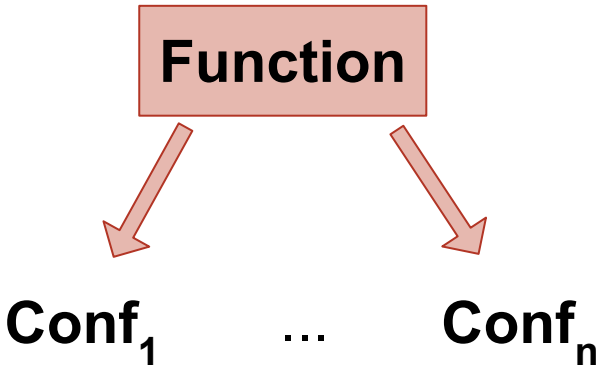


John Cavazos, J. Eliot B. Moss, Michael F. P. O'Boyle:
Hybrid Optimizations: Which Optimization Algorithm to Use?
CC 2006: 124-138

Example: Best Optimization (Program Scheduling)

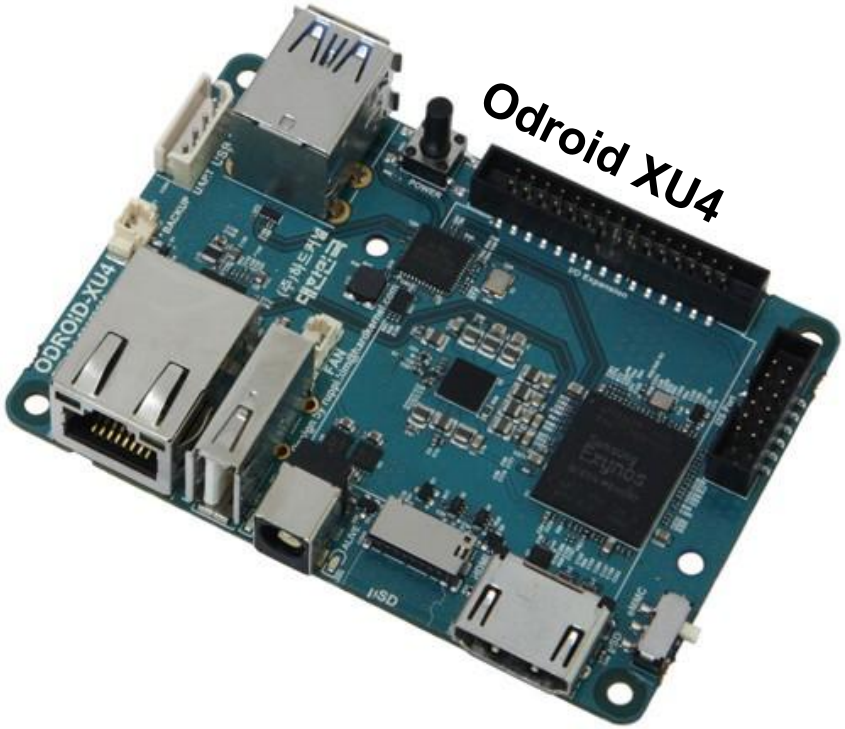


Example: Best Optimization (Program Scheduling)



Junio C. R. da Silva, Lorena Leão, Vinicius Petrucci, Abdoulaye Gamatié, Fernando M. Q. Pereira:
Mapping Computations in Heterogeneous Multicore Systems with Statistical Regression on Program Inputs. ACM Trans. Embed. Comput. Syst. 20(6): 112:1-112:35 (2021)

Example: Best Optimization (Program Scheduling)



Junio C. R. da Silva, Lorena Leão, Vinicius Petrucci, Abdoulaye Gamatié, Fernando M. Q. Pereira:
Mapping Computations in Heterogeneous Multicore Systems with Statistical Regression on Program Inputs. ACM Trans. Embed. Comput. Syst. 20(6): 112:1-112:35 (2021)

Example: Best Optimization (Program Scheduling)

big

0

1

2

3

Junio C. R. da Silva, Lorena Leão, Vinicius Petrucci, Abdoulaye Gamatié, Fernando M. Q. Pereira:
Mapping Computations in Heterogeneous Multicore Systems with Statistical Regression on Program Inputs. ACM Trans. Embed. Comput. Syst. 20(6): 112:1-112:35 (2021)

Example: Best Optimization (Program Scheduling)

big	LITTLE
0	0
1	1
2	2
3	3

Junio C. R. da Silva, Lorena Leão, Vinicius Petrucci, Abdoulaye Gamatié, Fernando M. Q. Pereira:
Mapping Computations in Heterogeneous Multicore Systems with Statistical Regression on Program Inputs. ACM Trans. Embed. Comput. Syst. 20(6): 112:1-112:35 (2021)

Example: Best Optimization (Program Scheduling)

big	Freq	LITTLE	Freq
0	200Khz	0	200Khz
1	300Khz	1	300Khz
2	...	2	...
3	1.9Ghz	3	1.7Ghz
	2.0Ghz		1.8Ghz

Junio C. R. da Silva, Lorena Leão, Vinicius Petrucci, Abdoulaye Gamatié, Fernando M. Q. Pereira:
Mapping Computations in Heterogeneous Multicore Systems with Statistical Regression on Program Inputs. ACM Trans. Embed. Comput. Syst. 20(6): 112:1-112:35 (2021)

Example: Best Optimization (Program Scheduling)

big	Freq	LITTLE	Freq
0	200Khz	0	200Khz
1	300Khz	1	300Khz
2	...	2	...
3	1.9Ghz	3	1.7Ghz
	2.0Ghz		1.8Ghz

Junio C. R. da Silva, Lorena Leão, Vinicius Petrucci, Abdoulaye Gamatié, Fernando M. Q. Pereira:
Mapping Computations in Heterogeneous Multicore Systems with Statistical Regression on Program Inputs. ACM Trans. Embed. Comput. Syst. 20(6): 112:1-112:35 (2021)

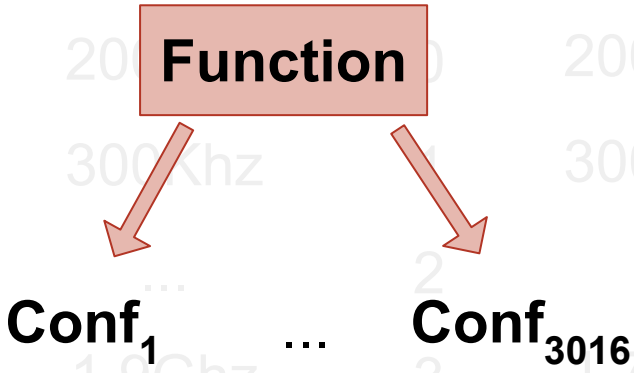
Example: Best Optimization (Program Scheduling)

big	Freq	LITTLE	Freq
0	200Khz	0	200Khz
1	300Khz	1	300Khz
2	...	2	...
3	1.9Ghz	3	1.7Ghz
	2.0Ghz		1.8Ghz

Configuration: 3 big cores at 1.8Ghz and 2 LITTLE cores at 1.0GHz

Junio C. R. da Silva, Lorena Leão, Vinicius Petrucci, Abdoulaye Gamatié, Fernando M. Q. Pereira:
Mapping Computations in Heterogeneous Multicore Systems with Statistical Regression on Program Inputs. ACM Trans. Embed. Comput. Syst. 20(6): 112:1-112:35 (2021)

Example: Best Optimization (Program Scheduling)



Configuration: 3 big cores at 1.8Ghz and 2 LITTLE cores at 1.0GHz

Junio C. R. da Silva, Lorena Leão, Vinicius Petrucci, Abdoulaye Gamatié, Fernando M. Q. Pereira:
Mapping Computations in Heterogeneous Multicore Systems with Statistical Regression on Program Inputs. ACM Trans. Embed. Comput. Syst. 20(6): 112:1-112:35 (2021)

Example: Best Optimization (Program Scheduling)

Junio C. R. da Silva, Lorena Leão, Vinicius Petrucci, Abdoulaye Gamatié, Fernando M. Q. Pereira:
Mapping Computations in Heterogeneous Multicore Systems with Statistical Regression on Program Inputs. ACM Trans. Embed. Comput. Syst. 20(6): 112:1-112:35 (2021)

Which Data to Use?

Which Data to Use?

- Program

Which Data to Use?

- Program
- Function

Which Data to Use?

- Program
- Function

Can you think about other granularities?

Which Data to Use?

- Program
- Function
- Instructions

Can you think about other granularities?

Which Data to Use?

- Program
- Function
- Instructions

What can you predict about instructions?

Can you think about other granularities?

Example: Silent Stores

Example: Silent Stores

- Store in memory a value already there

Example: Silent Stores

- Store in memory a value already there

How to guess if a store will be silent or not?

Example: Silent Stores

- Store in memory a value already there
- Regression

How to guess if a store will be silent or not?

Example: Silent Stores

- Store in memory a value already there

How to guess if a store will be silent or not?

Fernando Magno Quintão Pereira, Guilherme V. Leobas, Abdoulaye Gamatié:
Static Prediction of Silent Stores.
ACM Trans. Archit. Code Optim. 15(4): 44:1-44:26 (2019)

Example: Silent Stores

- Store in memory a value already there

```
a = a + b * c
```

How to guess if a store will be silent or not?

Fernando Magno Quintão Pereira, Guilherme V. Leobas, Abdoulaye Gamatié:
Static Prediction of Silent Stores.
ACM Trans. Archit. Code Optim. 15(4): 44:1-44:26 (2019)

Example: Silent Stores

- Store in memory a value already there

```
a = a + b * c
```

Fernando Magno Quintão Pereira, Guilherme V. Leobas, Abdoulaye Gamatié:
Static Prediction of Silent Stores.
ACM Trans. Archit. Code Optim. 15(4): 44:1-44:26 (2019)

Example: Silent Stores

- Store in memory a value already there

```
a = a + b * c
```

Can you think about stores that are unlikely to be silent?

Fernando Magno Quintão Pereira, Guilherme V. Leobas, Abdoulaye Gamatié:
Static Prediction of Silent Stores.
ACM Trans. Archit. Code Optim. 15(4): 44:1-44:26 (2019)

Example: Silent Stores

- Store in memory a value already there

```
a = ++a
```

Can you think about stores that are unlikely to be silent?

Fernando Magno Quintão Pereira, Guilherme V. Leobas, Abdoulaye Gamatié: *Static Prediction of Silent Stores.* ACM Trans. Archit. Code Optim. 15(4): 44:1-44:26 (2019)

Can you think about other properties to predict about instructions?

Example: Branch Prediction

Example: Branch Prediction

```
if (exp) {

    ...

} else {

    ...

}
```

Example: Branch Prediction

```
if (exp) {
    ...
} else {
    ...
}
```

Example: Branch Prediction

```
if (exp) {
    ...
} else {
    ...
}
```

Example: Branch Prediction

```
if (exp) {
    ...
} else {
    ...
}
```

Classification

Example: Branch Prediction

```
if (exp) {
    ...
} else {
    ...
}
```

Taken

Not Taken

Classification

Example: Branch Prediction

```
if (exp) {
    ...
} else {
    ...
}
```

Taken

Not Taken

Classification 

Brad Calder, Dirk Grunwald, M. Jones, Donald Lindsay, James Martin, Michael Mozer, B. Zorn:
Evidence-Based Static Branch Prediction Using Machine Learning.
ACM Trans. Program. Lang. Syst. 19(1): 188-222 (1997)

Example: Branch Prediction

```
if (exp) {
    ...
} else {
    ...
}
```

Taken

Not Taken

Classification

What would be the related regression problem?

Brad Calder, Dirk Grunwald, M. Jones, Donald Lindsay, James Martin, Michael Mozer, B. Zorn: *Evidence-Based Static Branch Prediction Using Machine Learning.* ACM Trans. Program. Lang. Syst. 19(1): 188-222 (1997)

Example: Branch Prediction

```
if (exp) {
    ...
} else {
    ...
}
```

87%

13%

Regression

Angélica Aparecida Moreira, Guilherme Ottoni, Fernando Magno Quintão Pereira:
VESPA: static profiling for binary optimization.
Proc. ACM Program. Lang. 5(OOPSLA): 1-28 (2021)

Just the Beginning

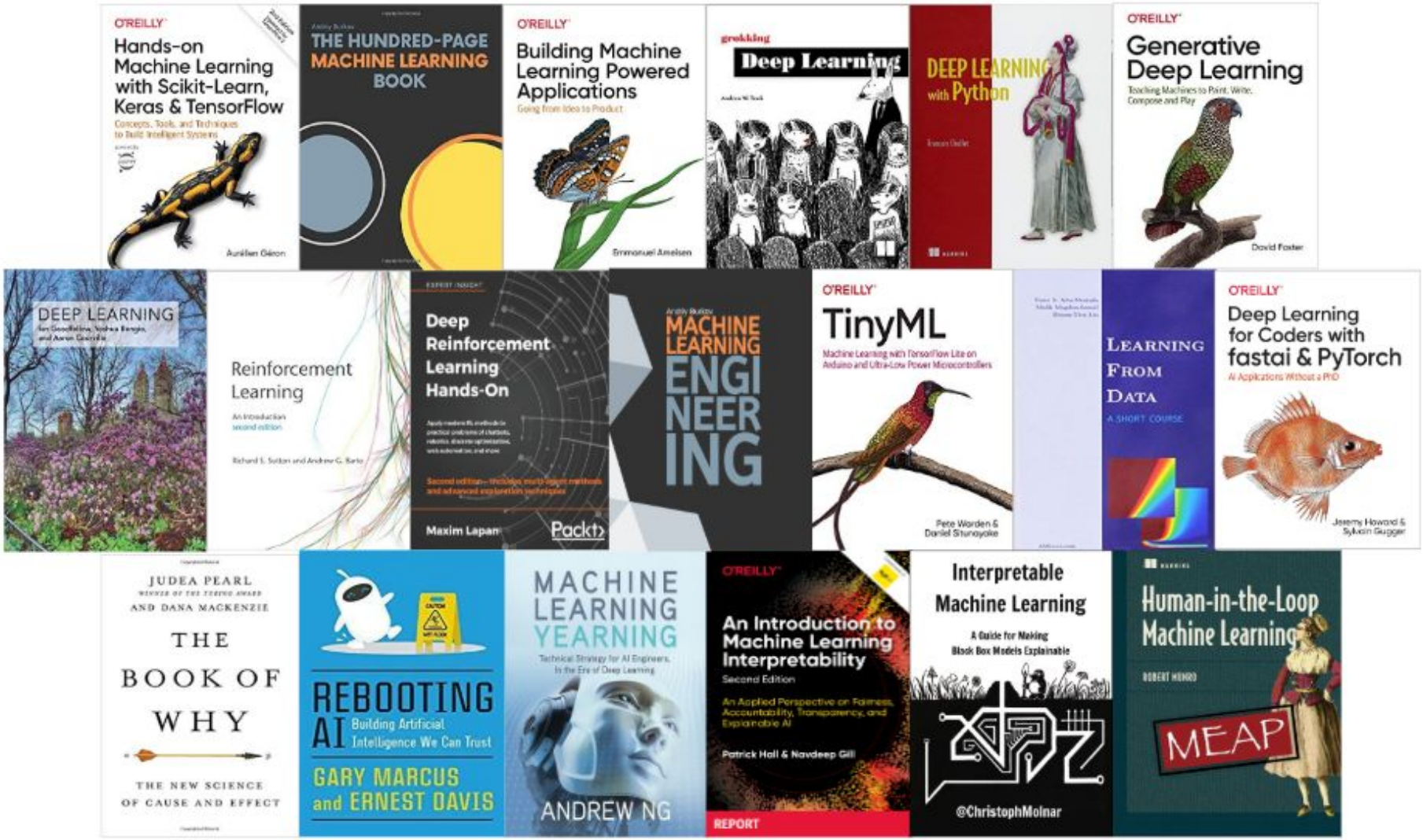
Compilers Laboratory

Department of Computer Science
Universidade Federal de Minas Gerais
Federal University of Minas Gerais, Brazil

Wrapping Up

DCC888

Just the Beginning



Just the Beginning

Amir H. Ashouri, William Killian, John Cavazos, Gianluca Palermo, Cristina Silvano:
A Survey on Compiler Autotuning using Machine Learning.
ACM Comput. Surv. 51(5): 96:1-96:42 (2019)

John Cavazos

John Cavazos

- University of Delaware

John Cavazos

- University of Delaware
- Nov. 3, 2018

John Cavazos

- University of Delaware
- Nov. 3, 2018
- 49

Compilers Laboratory

Department of Computer Science
Universidade Federal de Minas Gerais
Federal University of Minas Gerais, Brazil

Predictive Compilation

DCC888

Predicting the Impact of Optimizations

Predicting the Impact of Optimizations

John Cavazos, J. Eliot B. Moss, Michael F. P. O'Boyle:
Hybrid Optimizations: Which Optimization Algorithm to Use?
CC 2006: 124-138

Predicting the Impact of Optimizations

What is the improvement we can expect from a certain optimization?

John Cavazos, J. Eliot B. Moss, Michael F. P. O'Boyle:
Hybrid Optimizations: Which Optimization Algorithm to Use?
CC 2006: 124-138

Predicting the Impact of Optimizations

What is the improvement we can expect from a certain optimization?

John Cavazos, J. Eliot B. Moss, Michael F. P. O'Boyle:
Hybrid Optimizations: Which Optimization Algorithm to Use?
CC 2006: 124-138

Predicting the Impact of Optimizations

What is the code-size reduction that we can expect from a certain optimization?

Measuring Code Size

How to measure code size?

Measuring Code Size

How to measure code size?

What is code size?

Measuring Code Size

How to measure code size?

What is code size?

- Number of LLVM Instructions

Measuring Code Size

How to measure code size?

What is code size?

- Number of LLVM Instructions

```
clang -Xclang -disable-O0-optnone -S -emit-llvm file.c -o file.ll
```

Measuring Code Size

How to measure code size?

What is code size?

- Number of LLVM Instructions

```
clang -Xclang -disable-O0-optnone -S -emit-llvm file.c -o file.ll

opt -instcount -stats -disable-output file.ll
```

Measuring Code Size

How to measure code size?

What is code size?

Number of LLVM Instructions

```
clang -Xclang -disable-O0-optnone -S -emit-llvm file.c -o file.ll

opt -instcount -stats -disable-output file.ll
```

```
 2 instcount - Number of Add insts
 5 instcount - Number of Alloca insts
 ...
 1 instcount - Number of non-external functions
37 instcount - Number of instructions (of all types)
```

Measuring Code Size

How to measure code size?

What is code size?

- Size of the executable

Measuring Code Size

How to measure code size?

What is code size?

- Size of the executable

```
clang file.c -o file.exe

ls -la file.exe
```

```
-rwxr-xr-x  1 fernando  staff  16818 Mar  3 13:38 ./a.out
```

Predicting the Impact of Optimizations

Predicting the Impact of Optimizations

What is the code-size reduction that we can expect from **clang -Oz**?

Predicting the Impact of Optimizations

What is the code-size reduction that we can expect from **clang -Oz**?

Anderson Faustino da Silva, Bernardo N. B. de Lima, Fernando Magno Quintão Pereira: *Exploring the space of optimization sequences for code-size reduction: insights and tools.* CC 2021: 47-58

Predicting the Impact of Optimizations

- Median & Mean

Anderson Faustino da Silva, Bernardo N. B. de Lima, Fernando Magno Quintão Pereira: *Exploring the space of optimization sequences for code-size reduction: insights and tools.* CC 2021: 47-58

Predicting the Impact of Optimizations

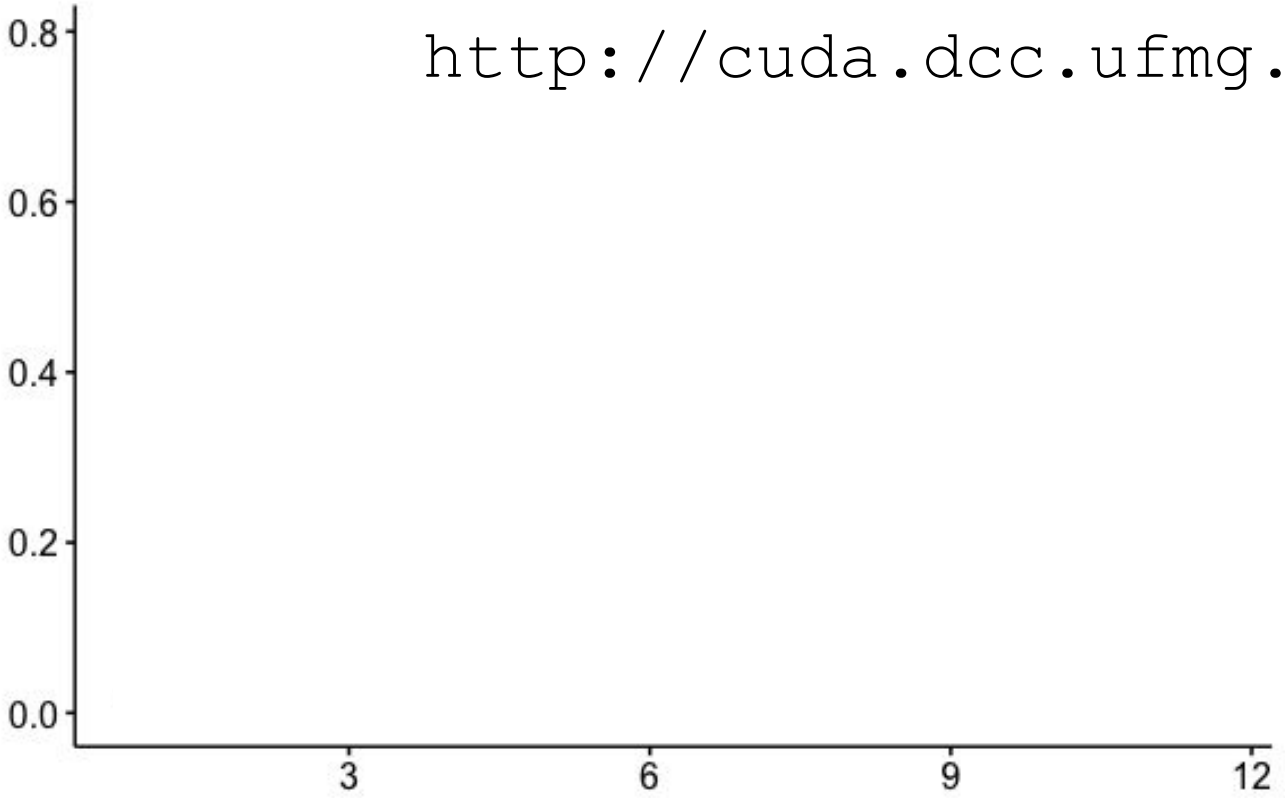
`http://cuda.dcc.ufmg.br/angha`

Predicting the Impact of Optimizations

`http://cuda.dcc.ufmg.br/angha`

A F Silva, B C Kind, J W S Magalhães, J N Rocha, B C F Guimarães, F M Q Pereira:
ANGHABENCH: A Suite with One Million Compilable C Benchmarks for Code-Size Reduction.
CGO 2021: 378-390

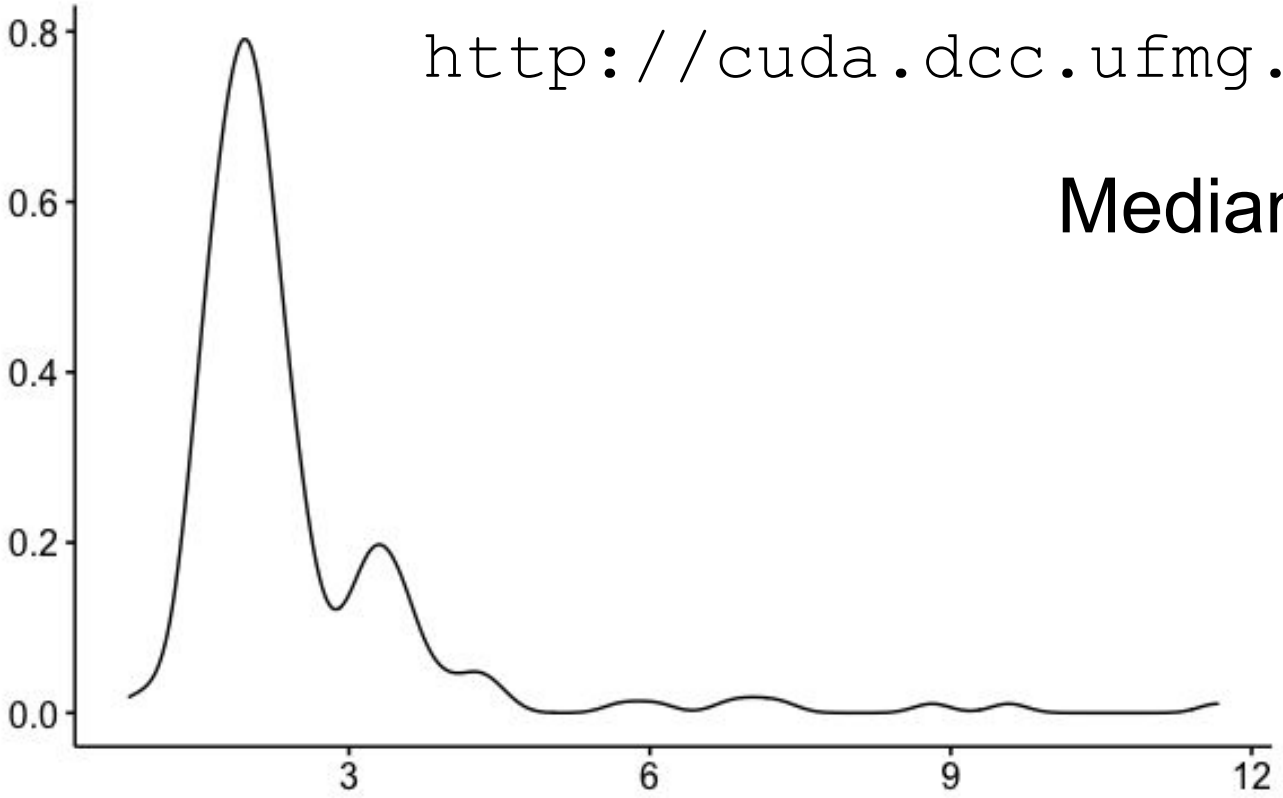
Predicting the Impact of Optimizations



http://cuda.dcc.ufmg.br/angha

A F Silva, B C Kind, J W S Magalhães, J N Rocha, B C F Guimarães, F M Q Pereira: *ANGHABENCH: A Suite with One Million Compilable C Benchmarks for Code-Size Reduction.* CGO 2021: 378-390

Predicting the Impact of Optimizations

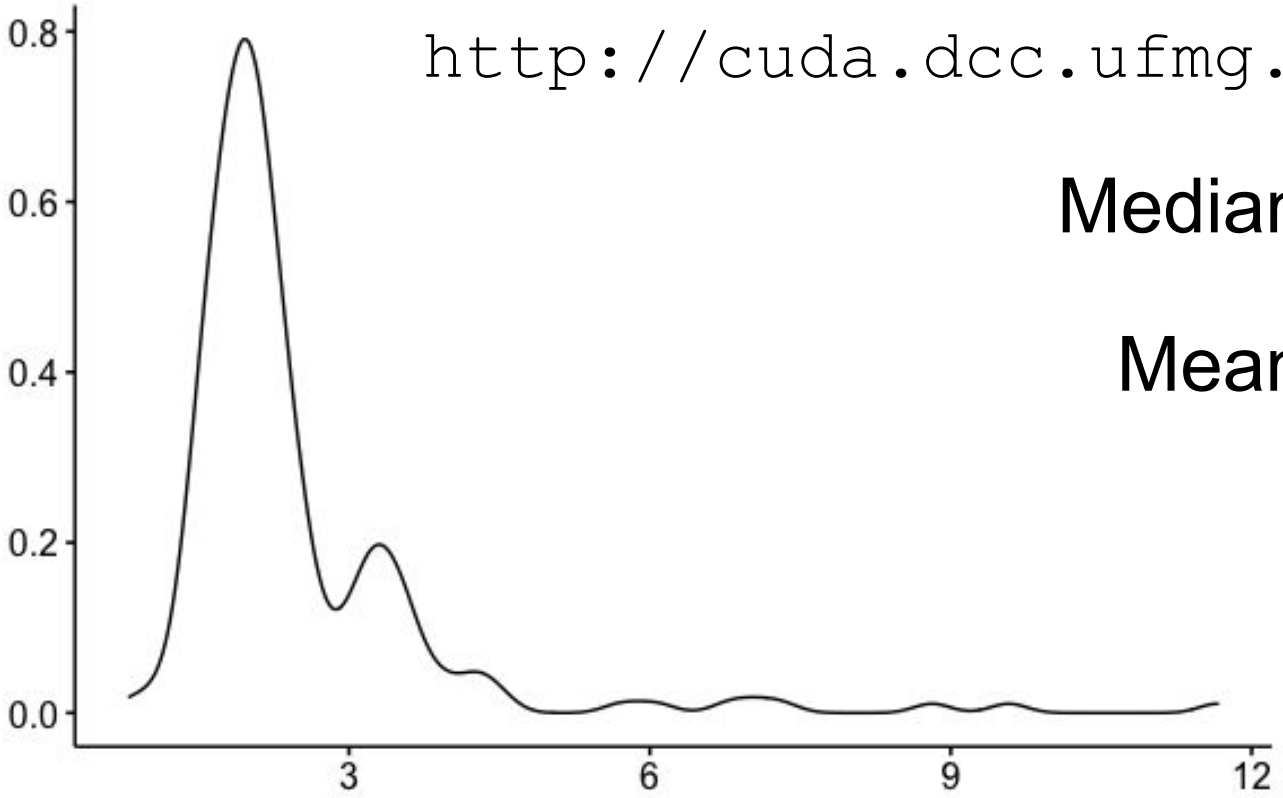


`http://cuda.dcc.ufmg.br/angha`

Median = 2.0917

A F Silva, B C Kind, J W S Magalhães, J N Rocha, B C F Guimarães, F M Q Pereira:
ANGHABENCH: A Suite with One Million Compilable C Benchmarks for Code-Size Reduction.
CGO 2021: 378-390

Predicting the Impact of Optimizations



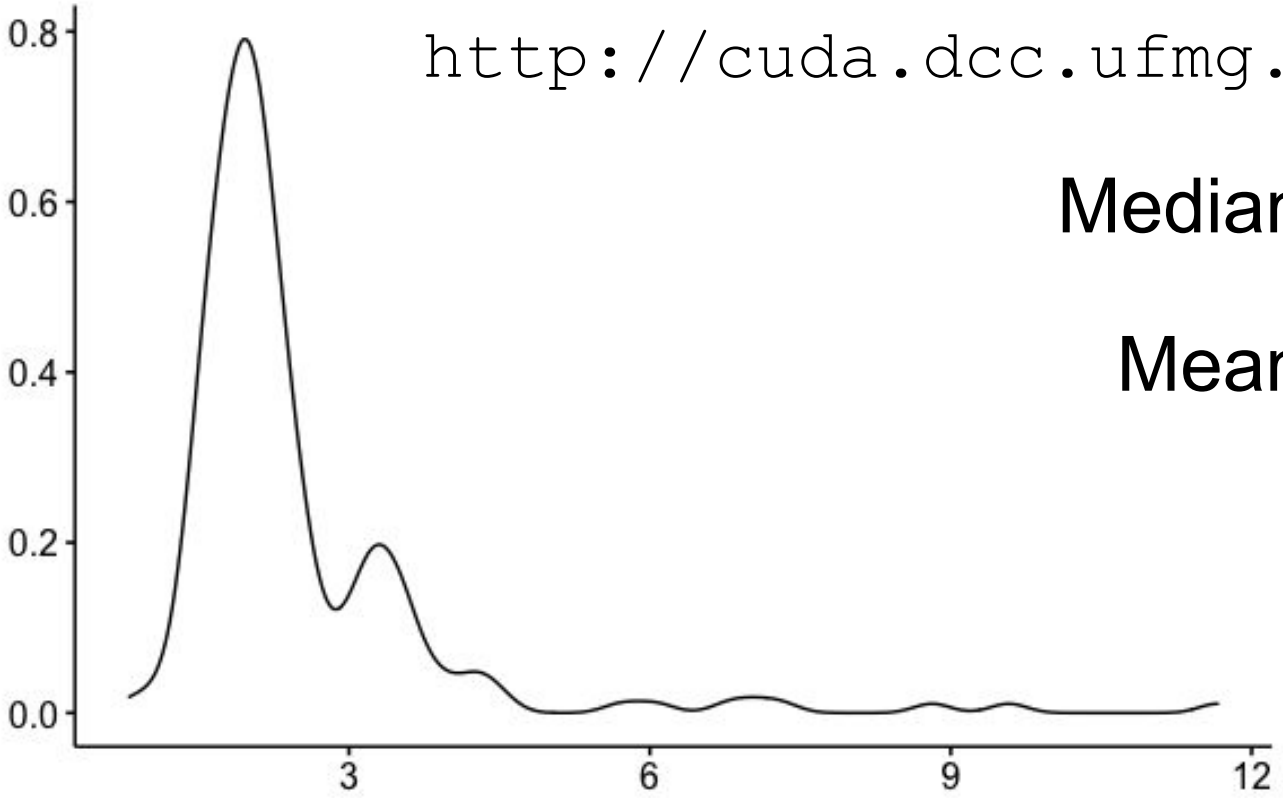
`http://cuda.dcc.ufmg.br/angha`

Median = 2.0917

Mean = 2.4734

A F Silva, B C Kind, J W S Magalhães, J N Rocha, B C F Guimarães, F M Q Pereira: *ANGHABENCH: A Suite with One Million Compilable C Benchmarks for Code-Size Reduction.* CGO 2021: 378-390

Predicting the Impact of Optimizations



`http://cuda.dcc.ufmg.br/angha`

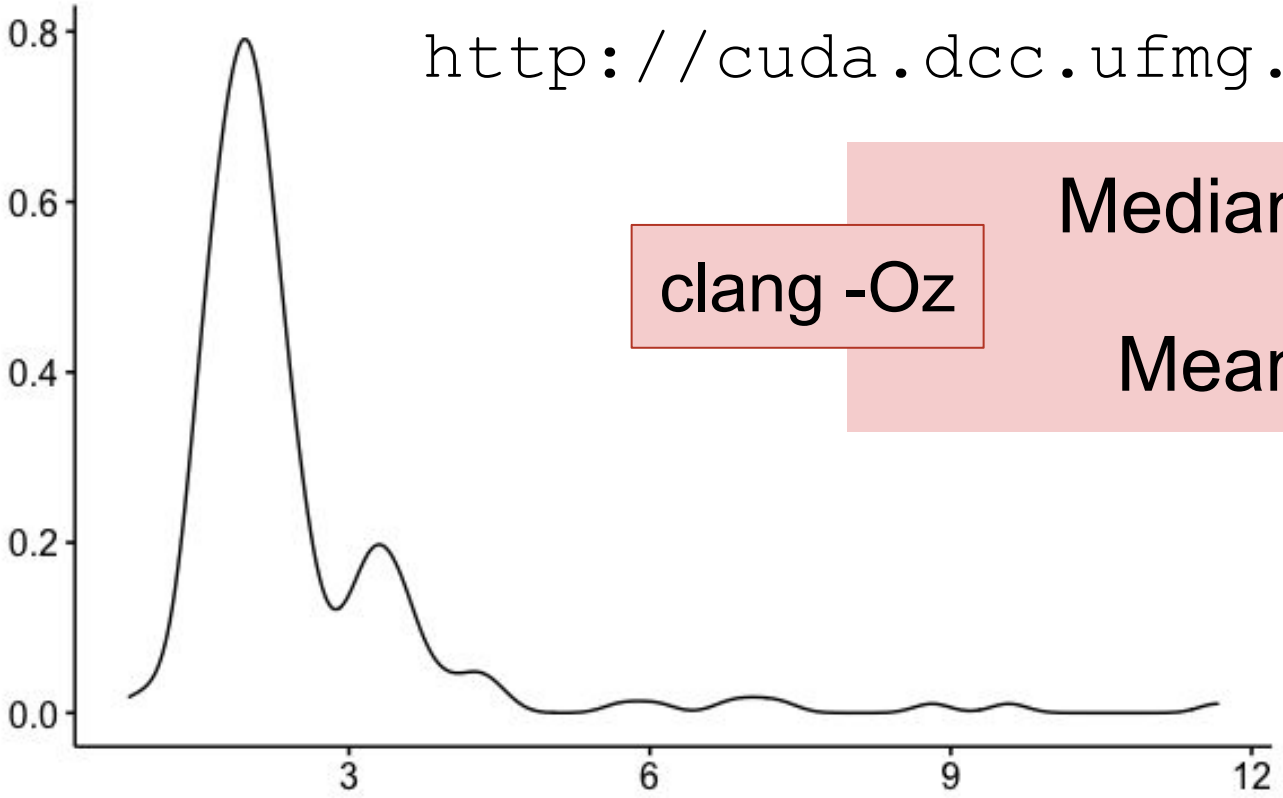
Median = 2.0917

Mean = 2.4734

LLVM IR

A F Silva, B C Kind, J W S Magalhães, J N Rocha, B C F Guimarães, F M Q Pereira:
ANGHABENCH: A Suite with One Million Compilable C Benchmarks for Code-Size Reduction.
CGO 2021: 378-390

Predicting the Impact of Optimizations



`http://cuda.dcc.ufmg.br/angha`

clang -Oz

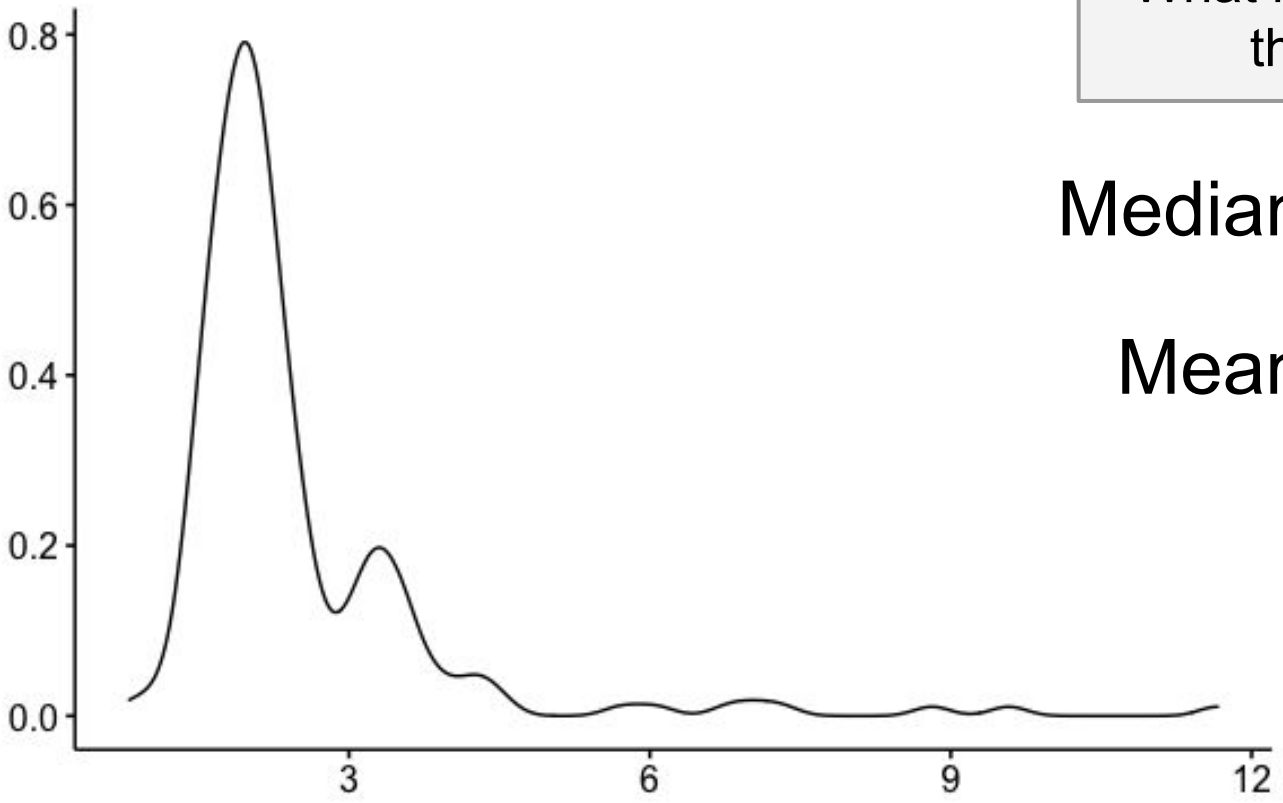
Median = 2.0917

Mean = 2.4734

LLVM IR

A F Silva, B C Kind, J W S Magalhães, J N Rocha, B C F Guimarães, F M Q Pereira:
ANGHABENCH: A Suite with One Million Compilable C Benchmarks for Code-Size Reduction.
CGO 2021: 378-390

Predicting the Impact of Optimizations



What is the problem with this approach?

Median = 2.0917

Mean = 2.4734

LLVM IR

Predictive Compilation

Predictive Compilation

Reiji Suda, Ken Naono, Keita Teranishi, John Cavazos:
Software Automatic Tuning: Concepts and State-of-the-Art Results. Software Automatic Tuning, From Concepts to State-of-the-Art Results 2010: 3-15

Predictive Compilation

Shun Long, Michael F. P. O'Boyle:
Adaptive Java optimisation using instance-based learning.
ICS 2004: 237-246

Reiji Suda, Ken Naono, Keita Teranishi, John Cavazos:
Software Automatic Tuning: Concepts and State-of-the-Art Results. Software Automatic Tuning, From Concepts to State-of-the-Art Results 2010: 3-15

Predictive Compilation

C Dubach, J Cavazos, B Franke, G Fursin, M O'Boyle, O Temam:
Fast compiler optimisation evaluation using code-feature based performance prediction.
Conf. Computing Frontiers 2007: 131-142

Shun Long, Michael F. P. O'Boyle:
Adaptive Java optimisation using instance-based learning.
ICS 2004: 237-246

Reiji Suda, Ken Naono, Keita Teranishi, John Cavazos:
Software Automatic Tuning: Concepts and State-of-the-Art Results. Software Automatic Tuning, From Concepts to State-of-the-Art Results 2010: 3-15

Predictive Compilation

P_A

P_B

Anderson Faustino da Silva, Bernardo N. B. de Lima, Fernando Magno Quintão Pereira: *Exploring the space of optimization sequences for code-size reduction: insights and tools.* CC 2021: 47-58

Predictive Compilation

P_A	Seq_5	-loop-rotate -sroa -correlated-propagation -indvars -gvn -tailcallelim -instcombine -jump-threading -reassociate -simplifycfg -instcombine -early-cse-memssa
P_B	Seq_4	-loop-vectorize -sroa -gvn -instcombine -simplifycfg -instcombine -licm -gvn -correlated-propagation -jump-threading -mldst-motion -early-cse-memssa -instcombine -simplifycfg -instsimplify

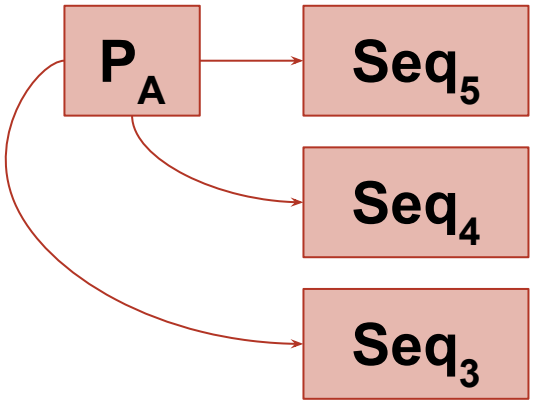
Anderson Faustino da Silva, Bernardo N. B. de Lima, Fernando Magno Quintão Pereira: *Exploring the space of optimization sequences for code-size reduction: insights and tools.* CC 2021: 47-58

Predictive Compilation

P_A	Seq_5	-loop-rotate -sroa -correlated-propagation -indvars -gvn -tailcallelim -instcombine -jump-threading -reassociate -simplifycfg -instcombine -early-cse-memssa
P_B	Seq_4	-loop-vectorize -sroa -gvn -instcombine -simplifycfg -instcombine -licm -gvn -correlated-propagation -jump-threading -mldst-motion -early-cse-memssa -instcombine -simplifycfg -instsimplify
	Seq_3	-mem2reg -early-cse-memssa -correlated-propagation -instcombine -reassociate -simplifycfg -early-cse-memssa -instcombine -licm -jump-threading -simplifycfg -dse -reassociate -early-cse-memssa -instcombine
	Seq_2	-sroa -early-cse-memssa -reassociate -instcombine -simplifycfg -licm -speculative-execution -jump-threading -early-cse-memssa -simplifycfg -instcombine -simplifycfg
	Seq_1	-mem2reg -jump-threading -instcombine -early-cse-memssa -jump-threading -licm -early-cse-memssa -sroa -simplifycfg -reassociate -instcombine -slp-vectorizer -early-cse-memssa

Anderson Faustino da Silva, Bernardo N. B. de Lima, Fernando Magno Quintão Pereira: *Exploring the space of optimization sequences for code-size reduction: insights and tools.* CC 2021: 47-58

Predictive Compilation



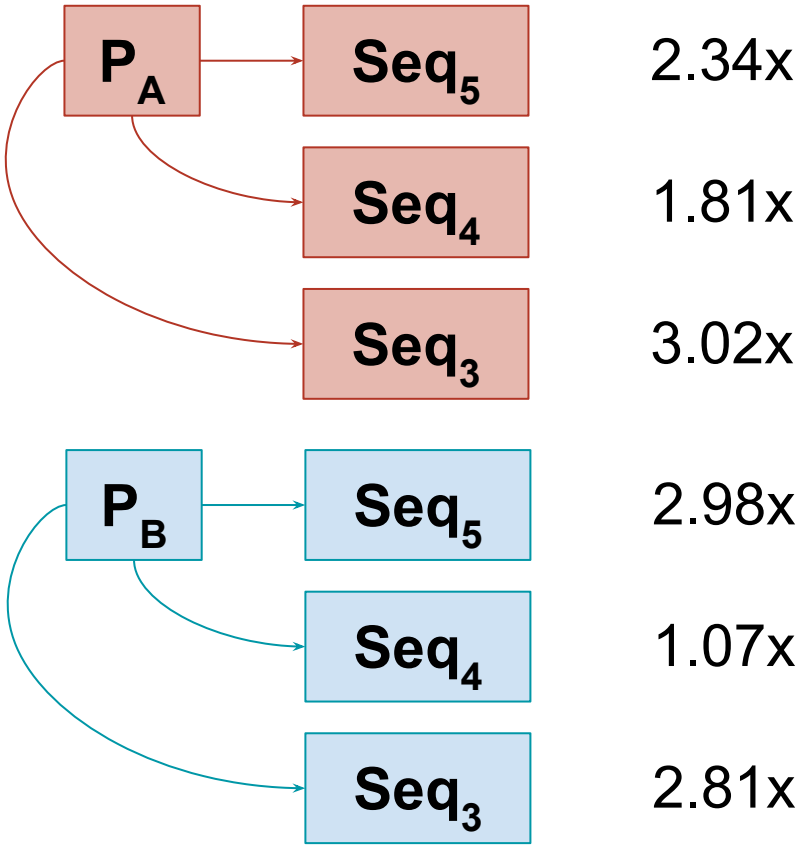
2.34x

1.81x

3.02x

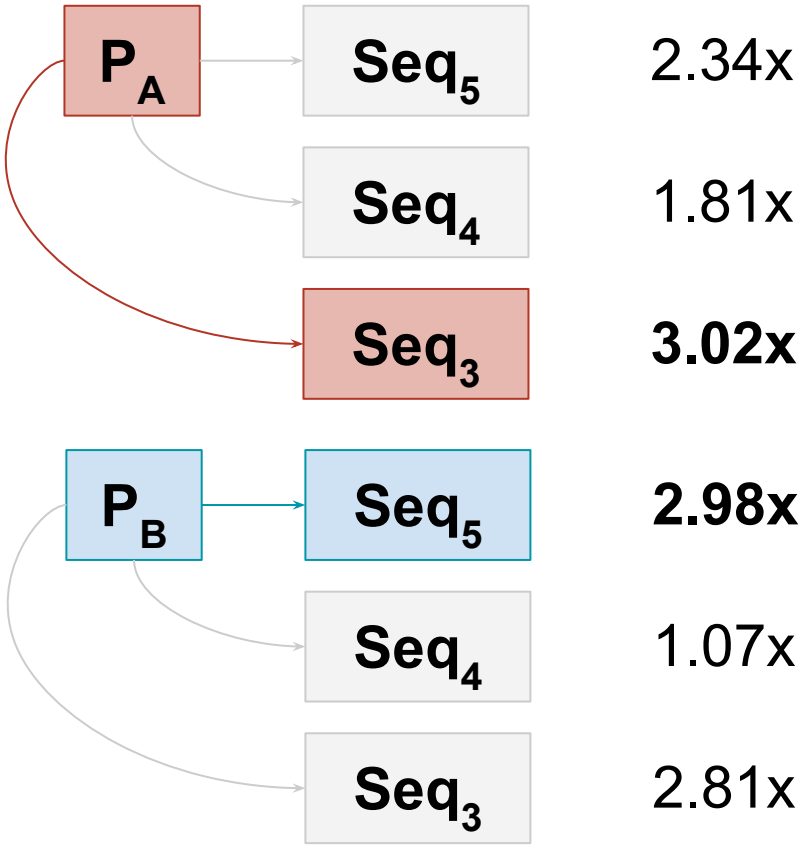
Anderson Faustino da Silva, Bernardo N. B. de Lima, Fernando Magno Quintão Pereira: *Exploring the space of optimization sequences for code-size reduction: insights and tools.* CC 2021: 47-58

Predictive Compilation



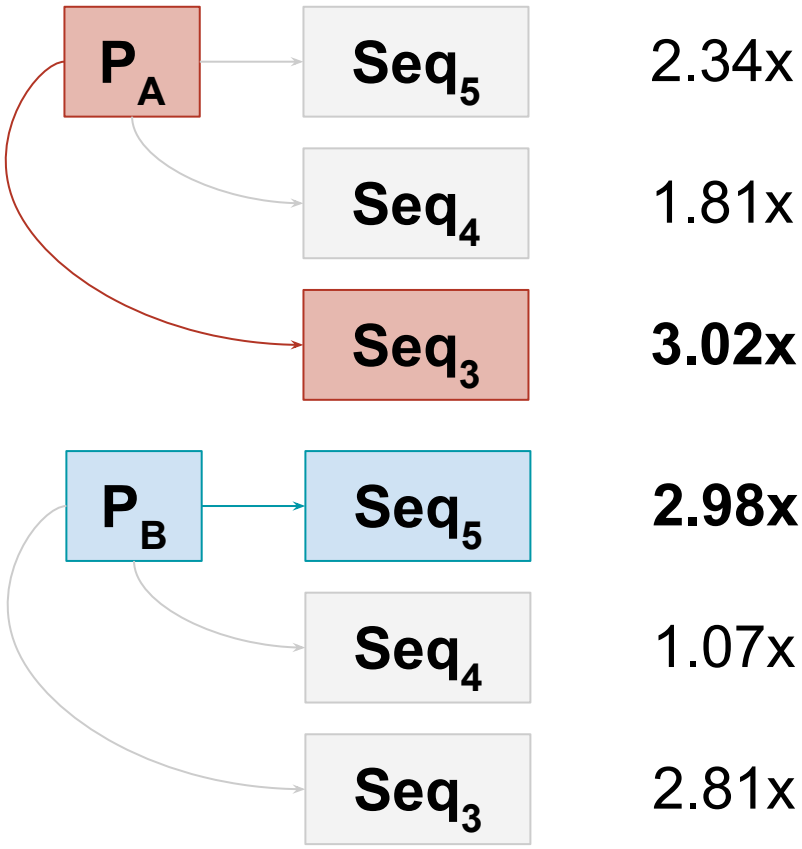
Anderson Faustino da Silva, Bernardo N. B. de Lima, Fernando Magno Quintão Pereira: *Exploring the space of optimization sequences for code-size reduction: insights and tools.* CC 2021: 47-58

Predictive Compilation



Anderson Faustino da Silva, Bernardo N. B. de Lima, Fernando Magno Quintão Pereira: *Exploring the space of optimization sequences for code-size reduction: insights and tools.* CC 2021: 47-58

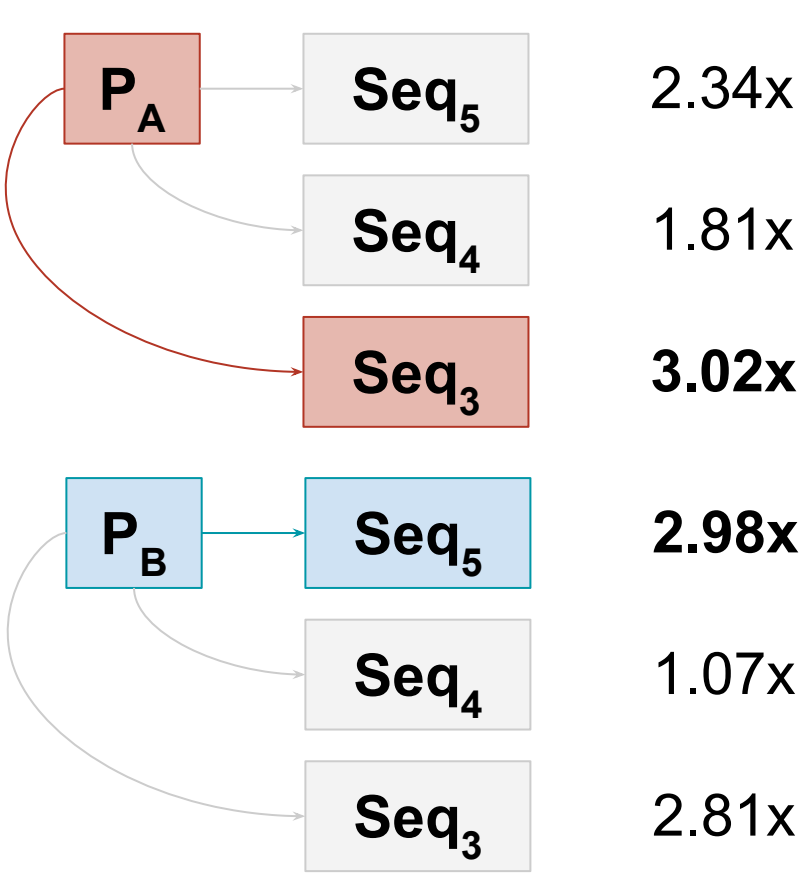
Predictive Compilation



Collect

Anderson Faustino da Silva, Bernardo N. B. de Lima, Fernando Magno Quintão Pereira: *Exploring the space of optimization sequences for code-size reduction: insights and tools.* CC 2021: 47-58

Predictive Compilation

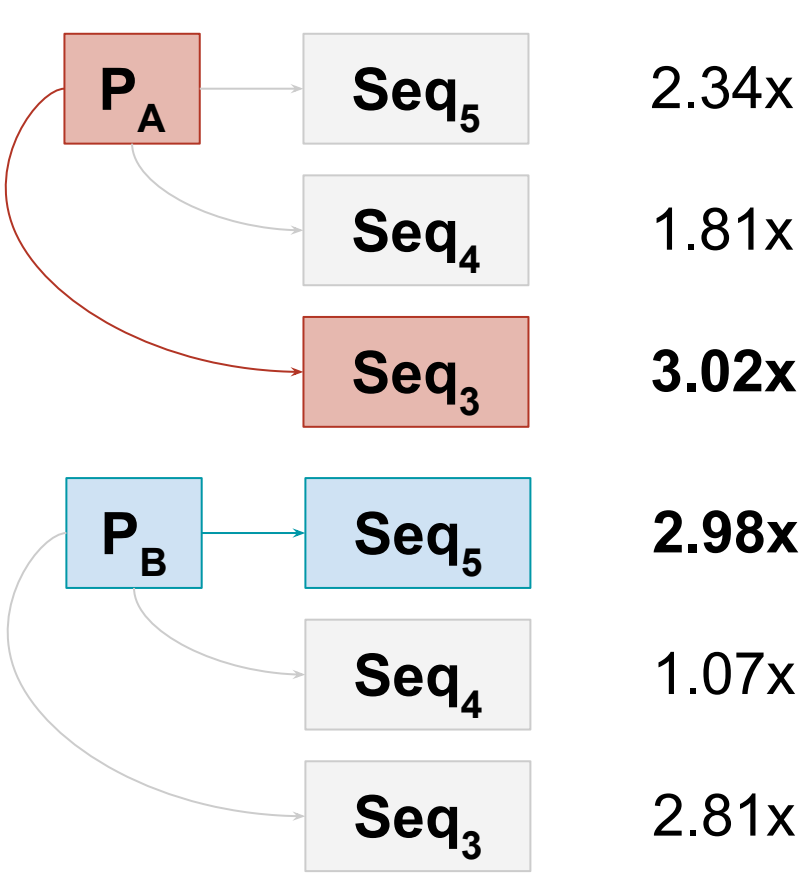


Collect

Train

Anderson Faustino da Silva, Bernardo N. B. de Lima, Fernando Magno Quintão Pereira: *Exploring the space of optimization sequences for code-size reduction: insights and tools.* CC 2021: 47-58

Predictive Compilation



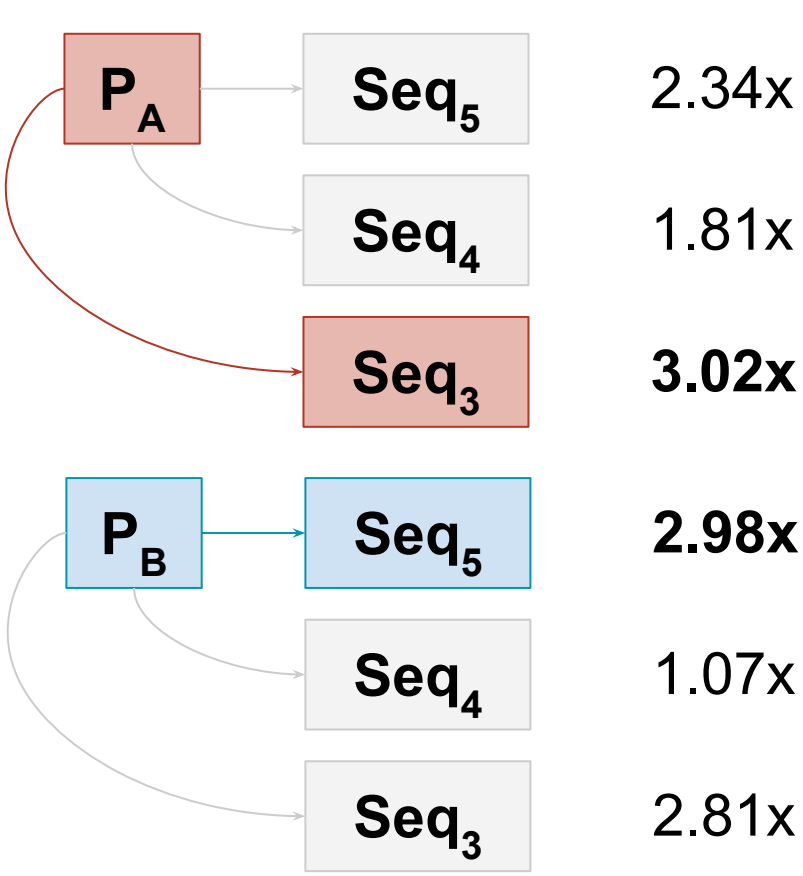
Collect

Train

Predict

Anderson Faustino da Silva, Bernardo N. B. de Lima, Fernando Magno Quintão Pereira: *Exploring the space of optimization sequences for code-size reduction: insights and tools.* CC 2021: 47-58

Predictive Compilation



P_U

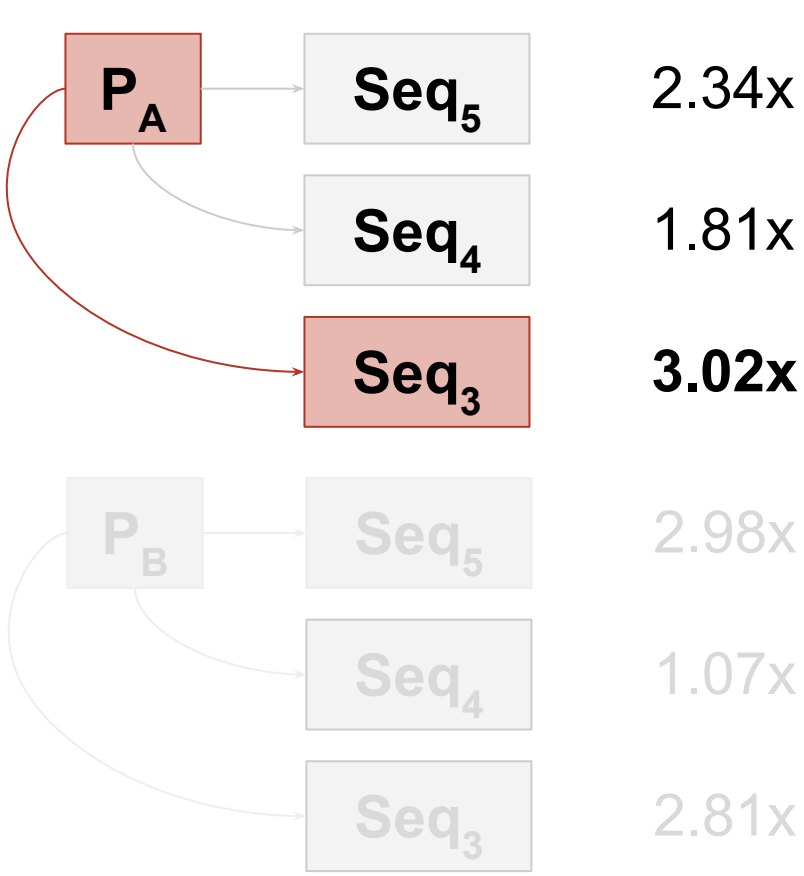
Collect

Train

Predict

Anderson Faustino da Silva, Bernardo N. B. de Lima, Fernando Magno Quintão Pereira: *Exploring the space of optimization sequences for code-size reduction: insights and tools.* CC 2021: 47-58

Predictive Compilation



P_U

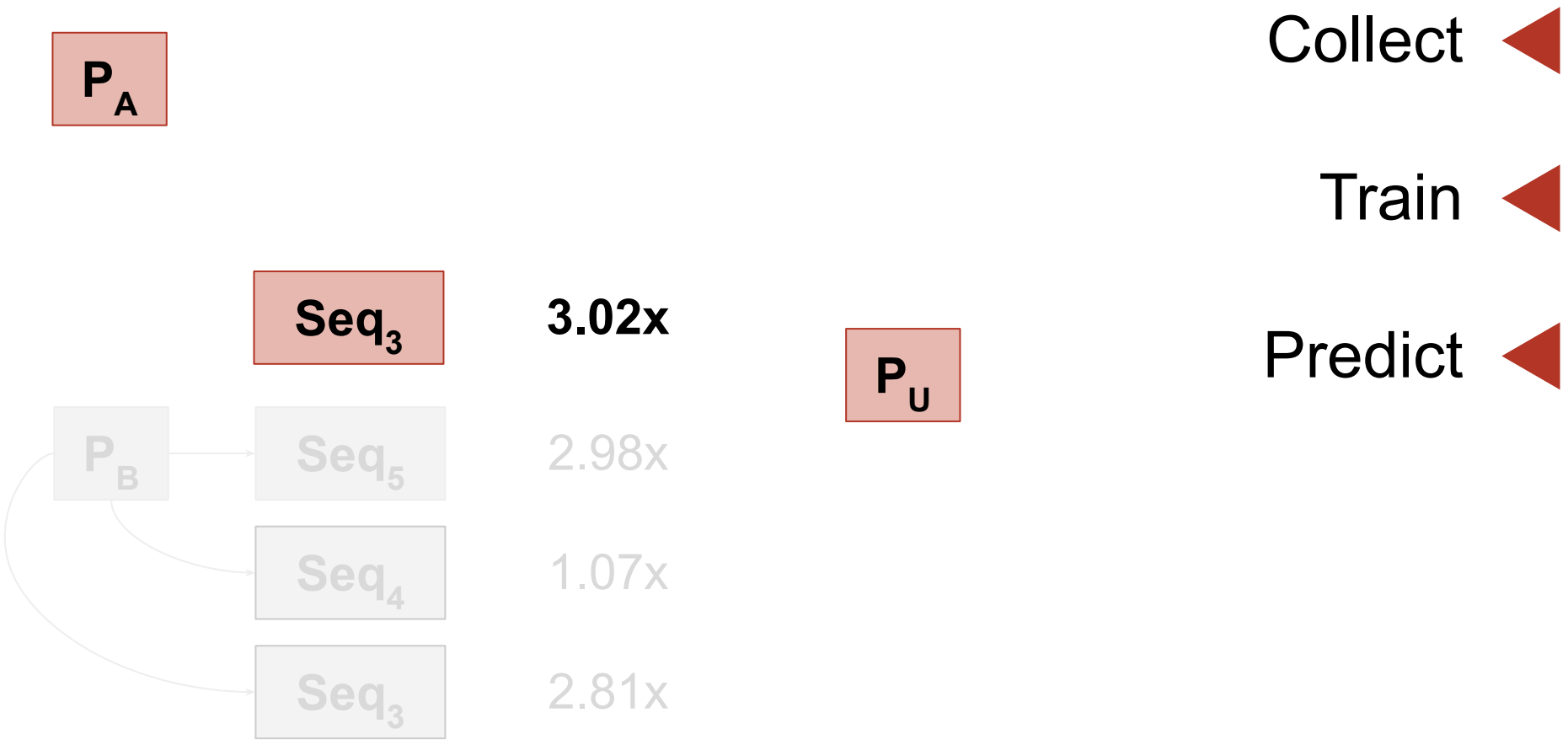
Collect

Train

Predict

Anderson Faustino da Silva, Bernardo N. B. de Lima, Fernando Magno Quintão Pereira: *Exploring the space of optimization sequences for code-size reduction: insights and tools.* CC 2021: 47-58

Predictive Compilation



Anderson Faustino da Silva, Bernardo N. B. de Lima, Fernando Magno Quintão Pereira: *Exploring the space of optimization sequences for code-size reduction: insights and tools.* CC 2021: 47-58

Predictive Compilation

Collect

Train

Predict

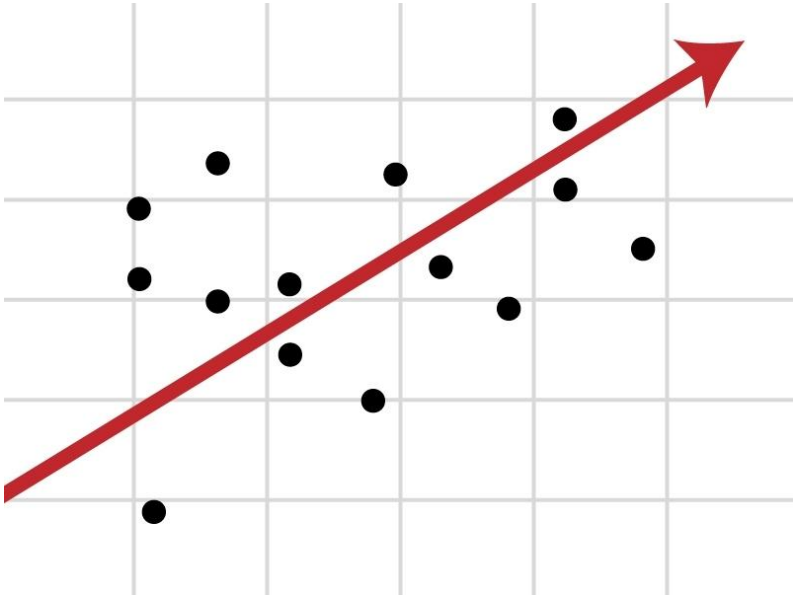
Anderson Faustino da Silva, Bernardo N. B. de Lima, Fernando Magno Quintão Pereira: *Exploring the space of optimization sequences for code-size reduction: insights and tools.* CC 2021: 47-58

Compilers Laboratory

Department of Computer Science
Universidade Federal de Minas Gerais
Federal University of Minas Gerais, Brazil

Regression

DCC888



Predictive Compilation

Predictive Compilation

- Estimate Code-Size Reduction

Predictive Compilation

- Estimate Code-Size Reduction

Anderson Faustino da Silva, Bernardo N. B. de Lima, Fernando Magno Quintão Pereira: *Exploring the space of optimization sequences for code-size reduction: insights and tools.* CC 2021: 47-58

Program Characteristics

Anderson Faustino da Silva, Bernardo N. B. de Lima, Fernando Magno Quintão Pereira: *Exploring the space of optimization sequences for code-size reduction: insights and tools.* CC 2021: 47-58

Program Characteristics Numeric Vectors

Anderson Faustino da Silva, Bernardo N. B. de Lima, Fernando Magno Quintão Pereira: *Exploring the space of optimization sequences for code-size reduction: insights and tools.* CC 2021: 47-58

Program Characteristics Numeric Vectors

P_A

P_B

Anderson Faustino da Silva, Bernardo N. B. de Lima, Fernando Magno Quintão Pereira: *Exploring the space of optimization sequences for code-size reduction: insights and tools.* CC 2021: 47-58

Program Characteristics Numeric Vectors

P_A

34	25	18	61
40	19	27	72
13	37	20	51
7	0	1	15

P_B

15	23	19	73
35	14	19	64
23	33	28	49
9	3	0	17

Anderson Faustino da Silva, Bernardo N. B. de Lima, Fernando Magno Quintão Pereira: *Exploring the space of optimization sequences for code-size reduction: insights and tools.* CC 2021: 47-58

Program Characteristics Numeric Vectors

P_A	34	25	18	61
P_B	15	23	19	73

Anderson Faustino da Silva, Bernardo N. B. de Lima, Fernando Magno Quintão Pereira: *Exploring the space of optimization sequences for code-size reduction: insights and tools.* CC 2021: 47-58

Program Characteristics Numeric Vectors

P_A	34
P_B	15

Anderson Faustino da Silva, Bernardo N. B. de Lima, Fernando Magno Quintão Pereira: *Exploring the space of optimization sequences for code-size reduction: insights and tools.* CC 2021: 47-58

Program Characteristics Numeric Vectors

P_A

34

How to transform a program in numbers?

P_B

15

Program Characteristics Numeric Vectors

P_A → 34

P_B → 15

How to transform a program in numbers?

Polynomial Time

Program Characteristics Numeric Vectors

P_A → 34

P_B → 15

How to transform a program in numbers?

Polynomial Time ◀

Deterministic ◀

Program Characteristics Numeric Vectors

P_A → 34

P_B → 15

How to transform a program in numbers?

- Polynomial Time
- Deterministic
- Collisions are Acceptable

Program Characteristics Numeric Vectors

P_A 34

P_B 15

How to transform a program in numbers?

Polynomial Time

Deterministic

Collisions are Acceptable

One way

Embedding Function

P_A

34

P_B

15

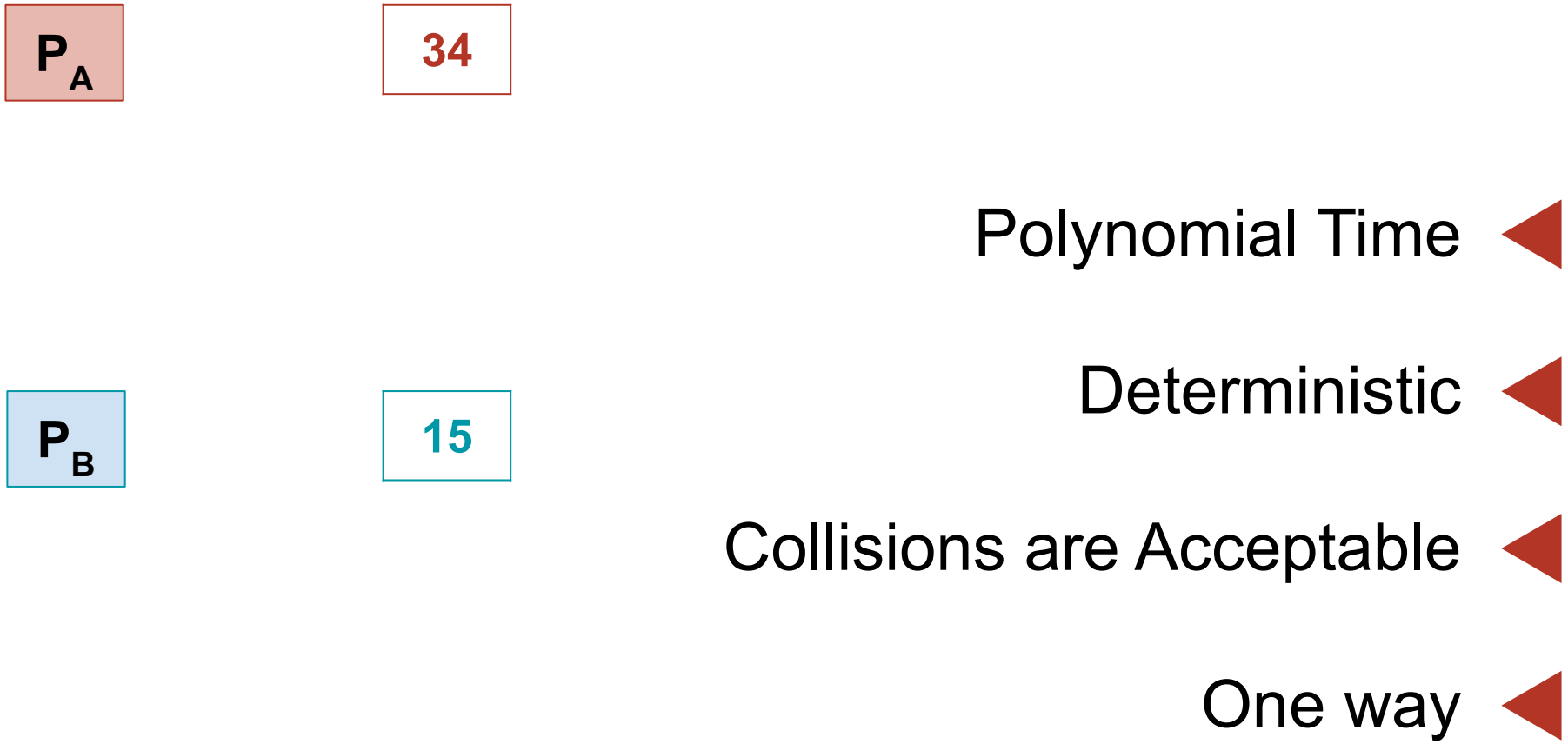
Polynomial Time

Deterministic

Collisions are Acceptable

One way

Embedding Function



Alexander Brauckmann, Andrés Goens, Sebastian Ertel, Jerónimo Castrillón:
Compiler-based graph representations for deep learning models of code.
CC 2020: 201-211

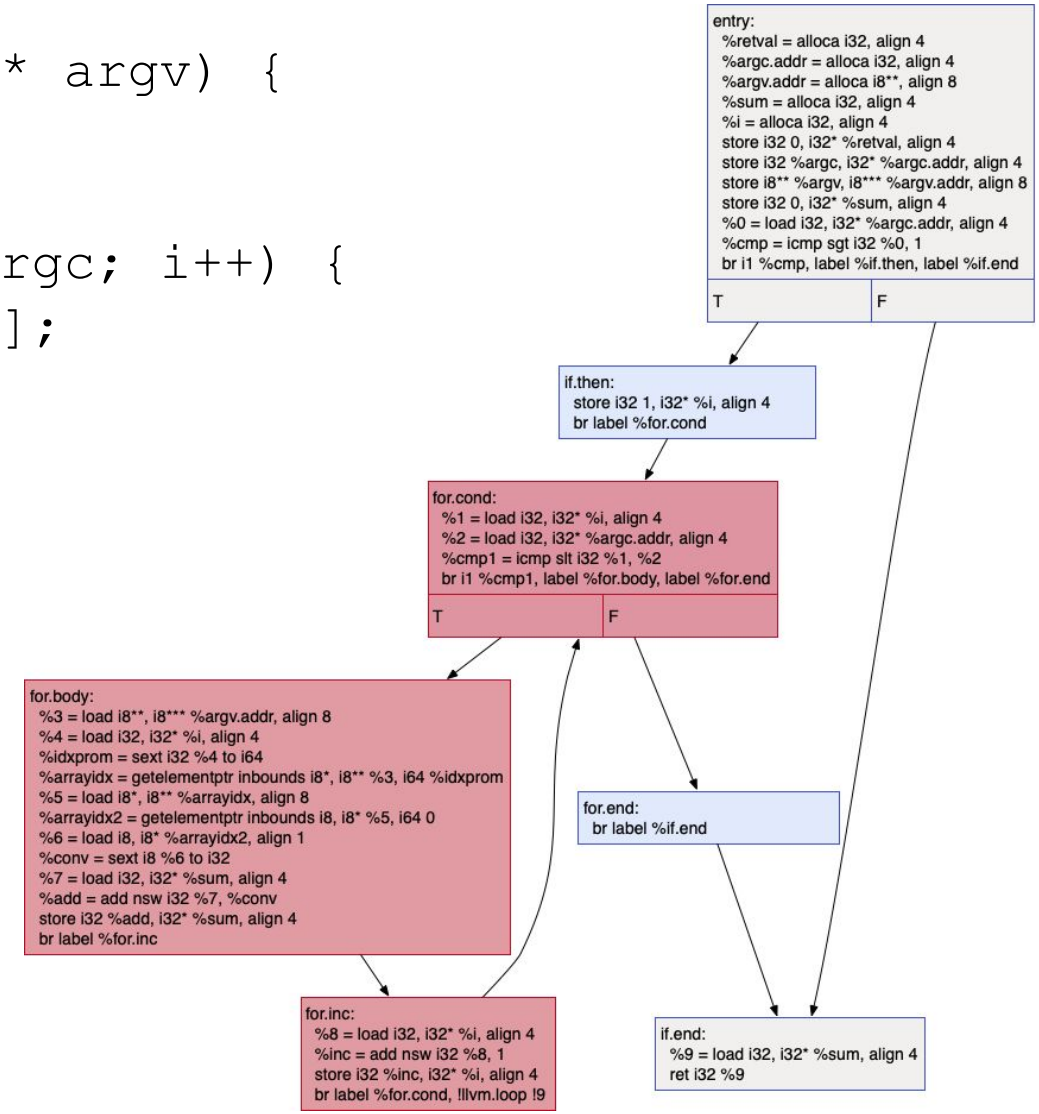
Histograms

Histograms

```
int main(int argc, char** argv) {
  int sum = 0;
  if (argc > 1) {
   for (int i = 1; i < argc; i++) {
   sum += (int)argv[i][0];
   }
  }
  return sum;
}
```

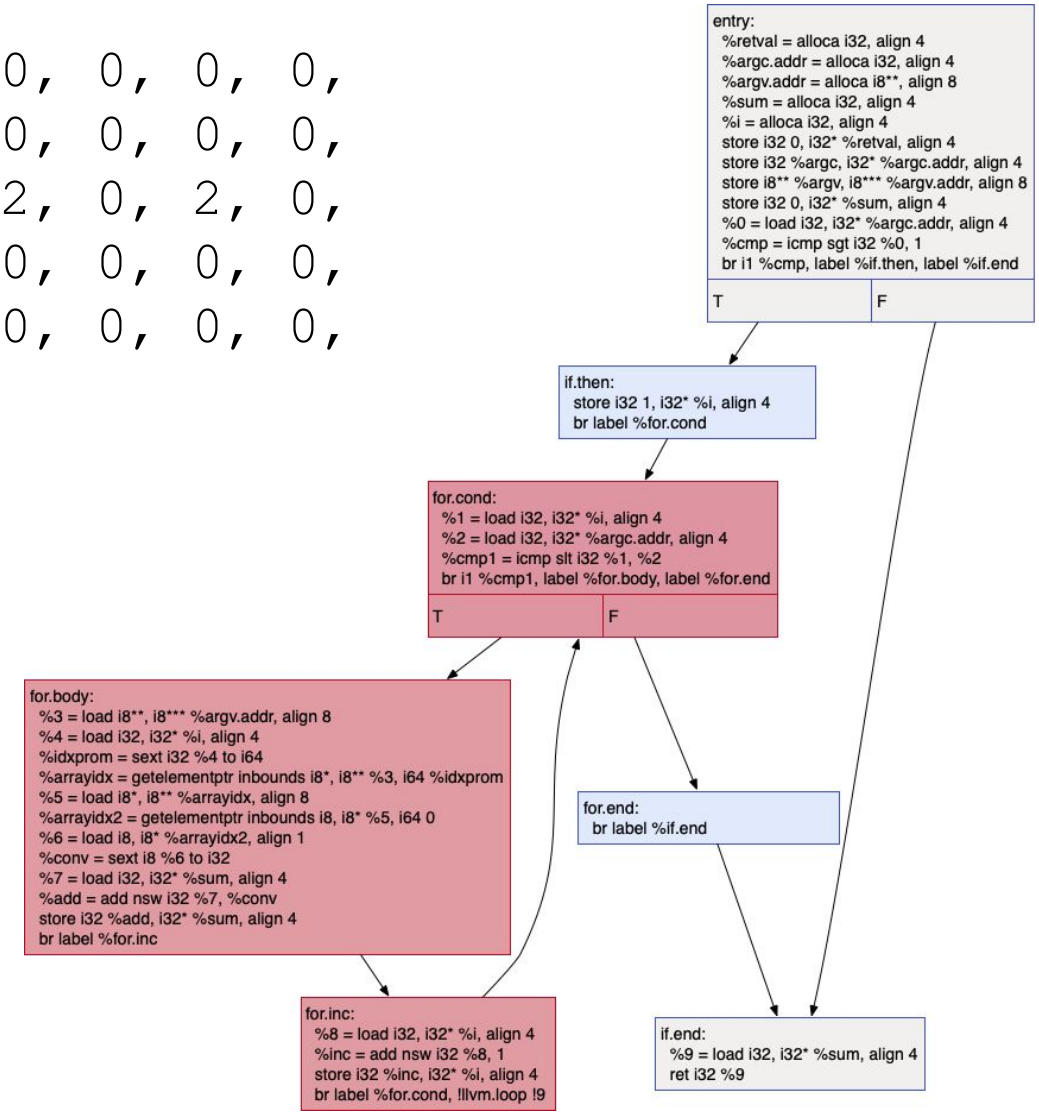
Histograms

```
int main(int argc, char** argv) {
   int sum = 0;
   if (argc > 1) {
    for (int i = 1; i < argc; i++) {
    sum += (int)argv[i][0];
    }
   }
   return sum;
}
```



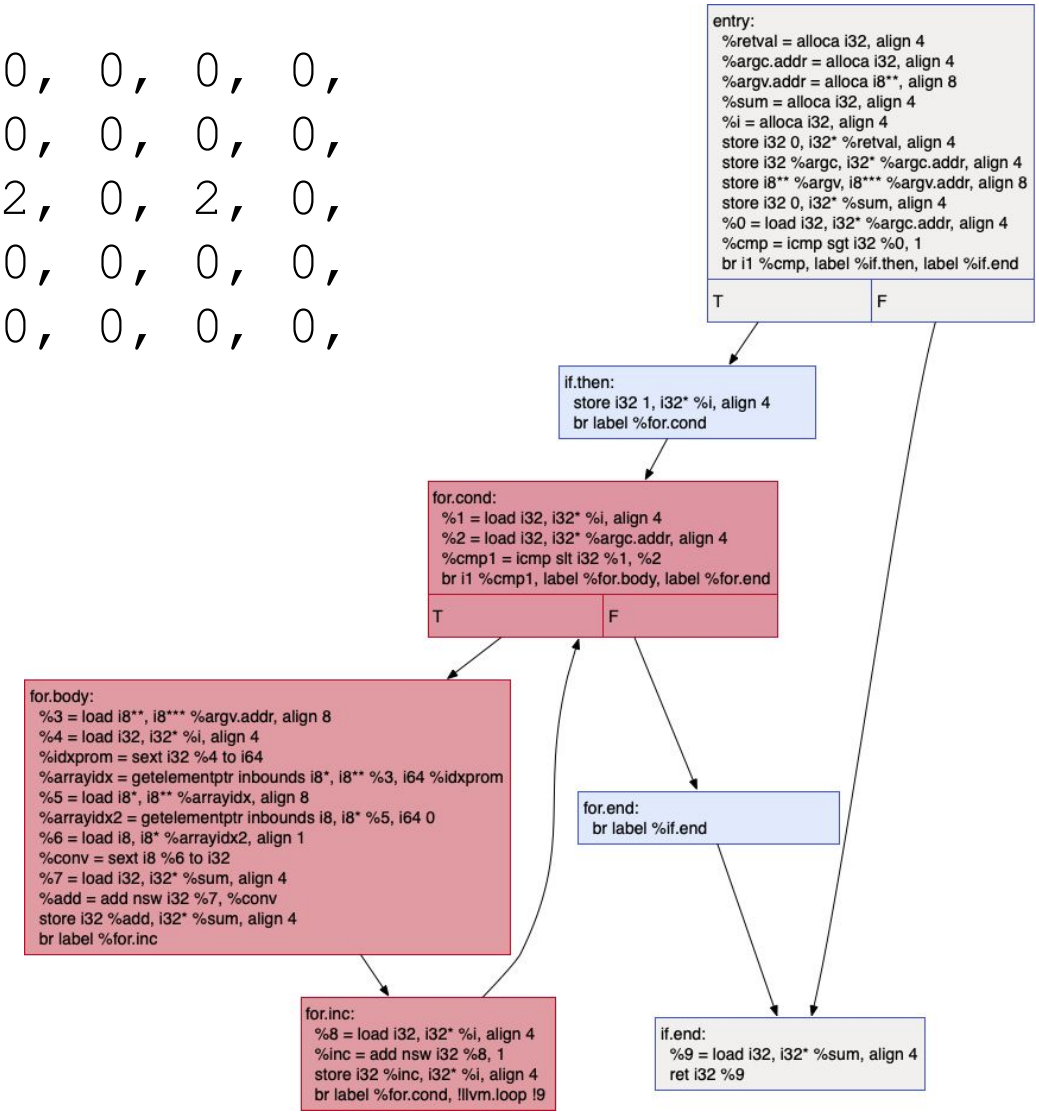
Histograms

```
0, 1, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 2, 0,
0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 2, 0, 3, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0
```



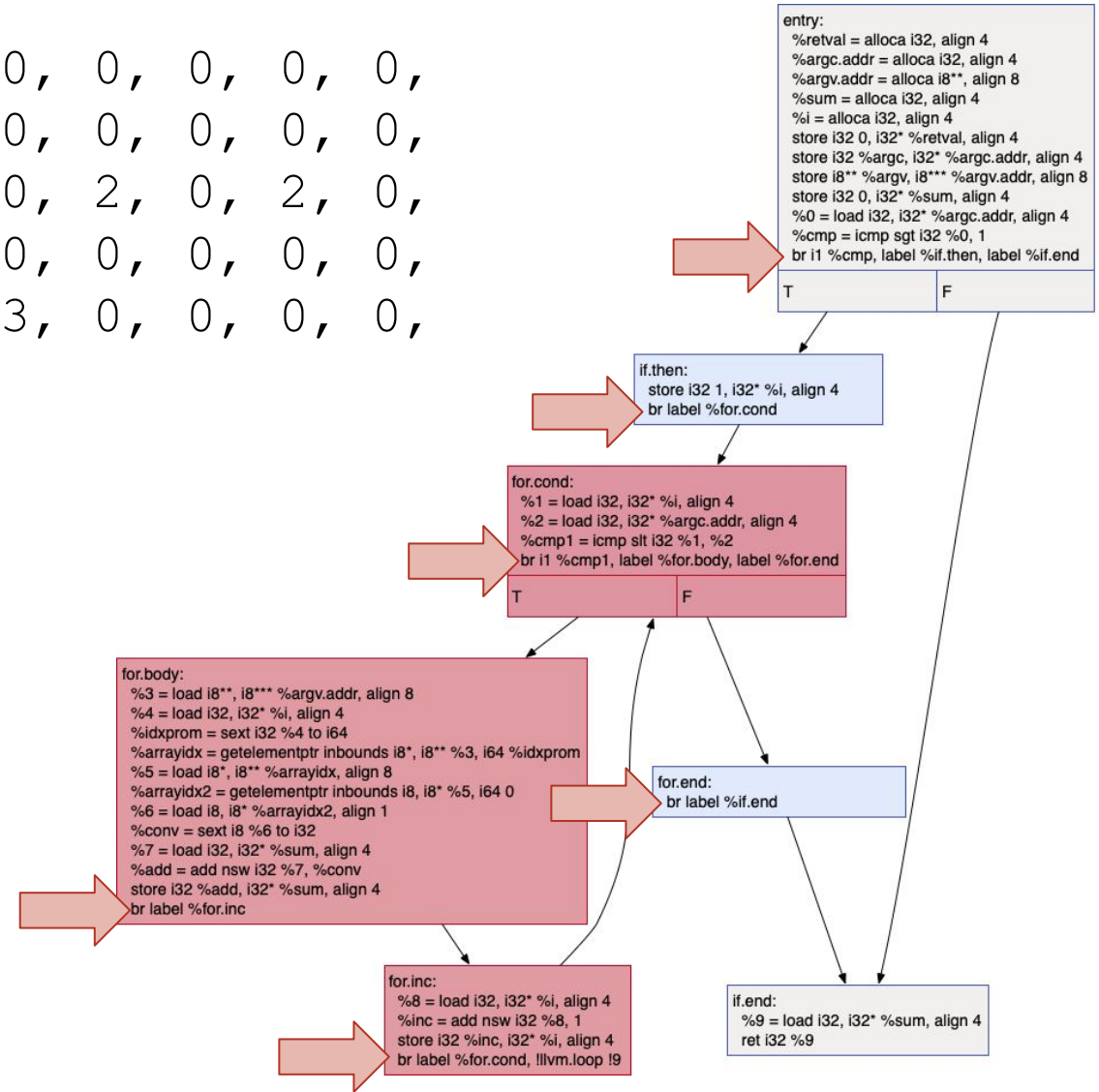
Histograms

```
0, 1, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 2, 0,
0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 2, 0, 3, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0
```



Histograms

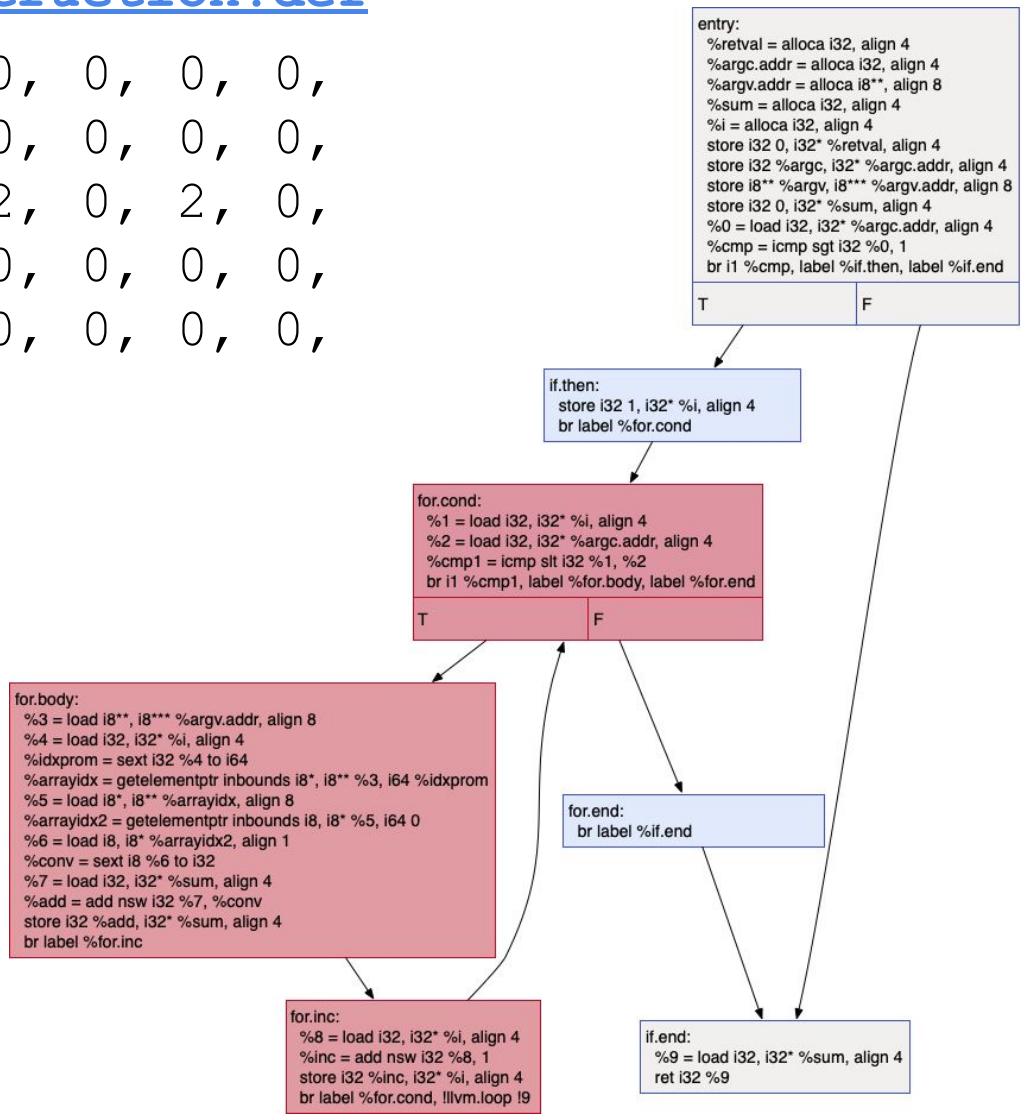
```
0, 1, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 2, 0,
0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 2, 0, 3, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0
```



Histograms

llvm/include/llvm/IR/Instruction.def

```
0, 1, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 2, 0,
0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 2, 0, 3, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0
```

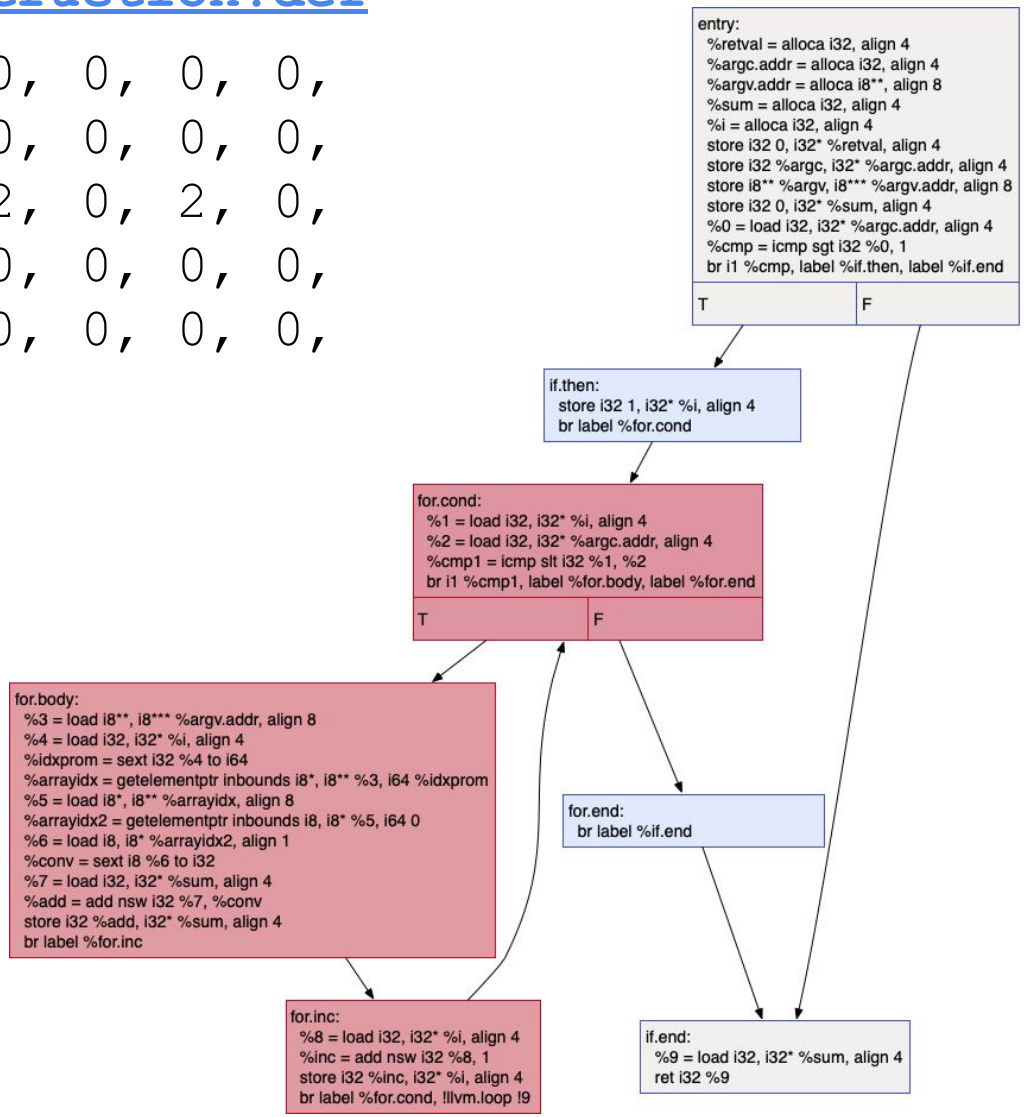


Histograms

llvm/include/llvm/IR/Instruction.def

```
0, 1, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 2, 0,
0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 2, 0, 3, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0
```

How to use the histogram to predict the best sequence of optimizations for a program?

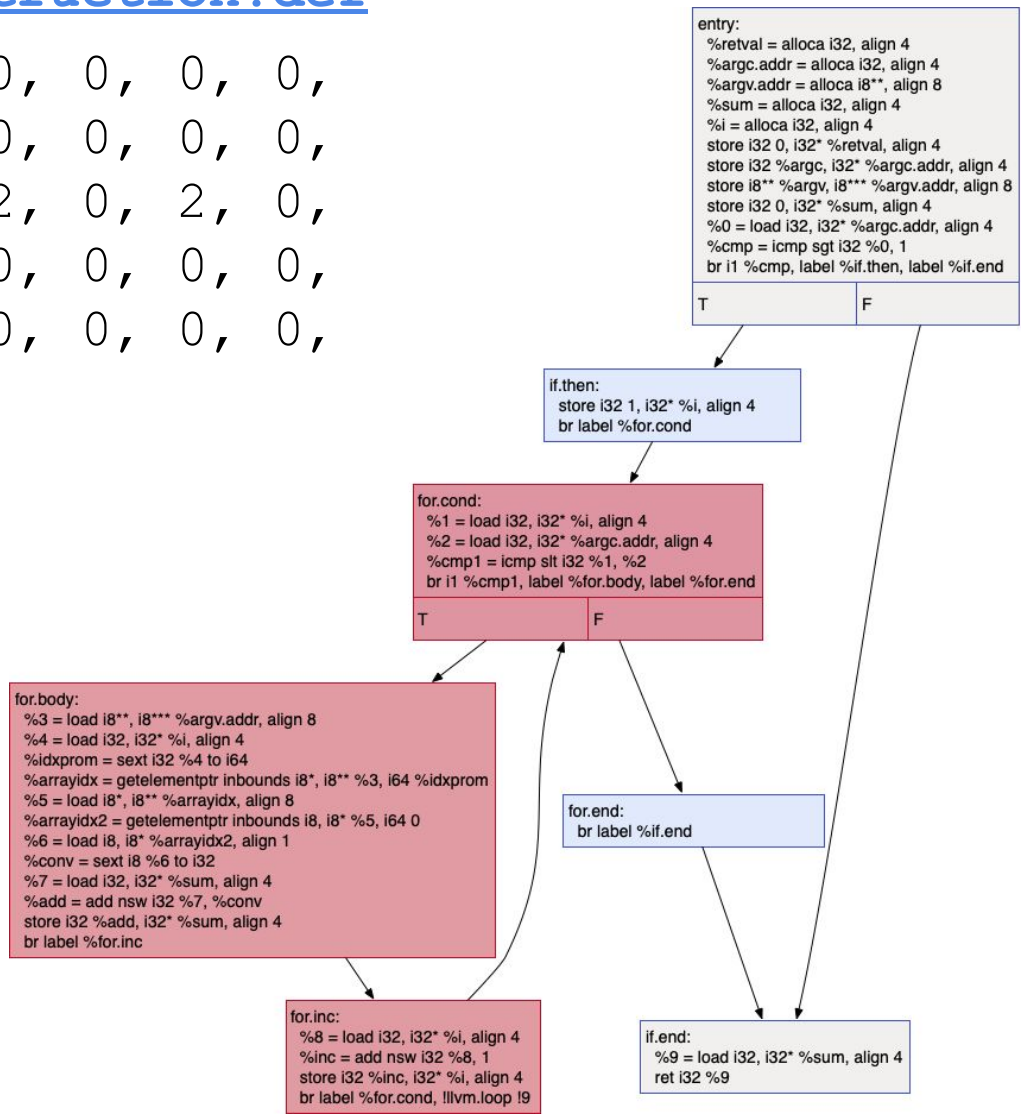


Histograms

llvm/include/llvm/IR/Instruction.def

```
0, 1, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 2, 0,
0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 2, 0, 3, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0
```

How to use the histogram to predict the effect of a sequence of optimizations on a program?



Input:

```
0, 1, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 2, 0,
0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 2, 0, 3, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0
```

-mem2reg -jump-threading -instcombine -early-cse-memssa -jump-threading -licm -early-cse-memssa -sroa -simplifycfg -reassociate -instcombine -slp-vectorizer -early-cse-memssa

Input:

```
0, 1, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 2, 0,
0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 2, 0, 3, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0
```

-mem2reg -jump-threading -instcombine -early-cse-memssa -jump-threading -licm -early-cse-memssa -sroa -simplifycfg -reassociate -instcombine -slp-vectorizer -early-cse-memssa

Output:

```
13 instructions
```

Input:

```
0, 1, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 2, 0,
0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 2, 0, 3, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0
```

Br Switch FAdd FSub Or Xor

Store GetElementPtr FPToUI

Feature Engineering

```
0, 1, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 2, 0,
0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 2, 0, 3, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0
```

Br Switch FAdd FSub Or Xor

Store GetElementPtr FPToUI

Feature Engineering

```
0, 1, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 2, 0,
0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 2, 0, 3, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0
```

Embedding: `(6, 0, 2, 0, 0, 0, 0, 2, 0)`

Br Switch FAdd FSub Or Xor

Store GetElementPtr FPToUI

Feature Engineering

```
0, 1, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 2, 0,
0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 2, 0, 3, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0
```

Embedding: `(6, 0, 2, 0, 0, 0, 0, 2, 0)`

$$N_{seq} = C_{br}Br + C_{switch}Switch + C_{FAdd}FAdd + C_{FSub}FSub + C_{Or}Or + C_{Xor}Xor + C_{Store}Store + C_{GetElementPtr}GetElementPtr + C_{FPToUI}FPToUI + C$$

Feature Engineering

```
0, 1, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 2, 0,
0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 2, 0, 3, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0
```

Embedding: `(6, 0, 2, 0, 0, 0, 0, 2, 0)`

N_{seq}

Linear Model

```
0, 1, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 2, 0,
0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 2, 0, 3, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0
```

Embedding: `(6, 0, 2, 0, 0, 0, 0, 2, 0)`

$$N_{seq} = C_{br}Br + C_{switch}Switch + C_{FAdd}FAdd + C_{FSub}FSub + C_{Or}Or + C_{Xor}Xor + C_{Store}Store + C_{GetElementPtr}GetElementPtr + C_{FPToUI}FPToUI + C$$

Linear Model

```
0, 1, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 2, 0,
0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 2, 0, 3, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0
```

Embedding: `(6, 0, 2, 0, 0, 0, 0, 2, 0)`

$$N_{seq} = C_{br}\,Br + C_{switch}\,Switch + C_{FAdd}\,FAdd + C_{FSub}\,FSub + C_{Or}\,Or + C_{Xor}\,Xor + C_{Store}\,Store + C_{GetElementPtr}\,GetElementPtr + C_{FPToUI}\,FPToUI + C$$

Linear Regression

$$N_{seq} = C_{br}Br + C_{switch}Switch + C_{FAdd}FAdd + C_{FSub}FSub + C_{Or}Or + C_{Xor}Xor + C_{Store}Store + C_{GetElementPtr}GetElementPtr + C_{FPToUI}FPToUI + C$$

Linear Regression

And how do we find all these constants?

$$N_{seq} = C_{br}Br + C_{switch}Switch + C_{FAdd}FAdd + C_{FSub}FSub + C_{Or}Or + C_{Xor}Xor + C_{Store}Store + C_{GetElementPtr}GetElementPtr + C_{FPToUI}FPToUI + C$$

Training!

$$N_{seq} = C_{br}Br + C_{switch}Switch + C_{FAdd}FAdd + C_{FSub}FSub + C_{Or}Or + C_{Xor}Xor$$
$$+ C_{Store}Store + C_{GetElementPtr}GetElementPtr + C_{FPToUI}FPToUI + C$$

Training!

- Data

$$N_{seq} = C_{br}Br + C_{switch}Switch + C_{FAdd}FAdd + C_{FSub}FSub + C_{Or}Or + C_{Xor}Xor + C_{Store}Store + C_{GetElementPtr}GetElementPtr + C_{FPToUI}FPToUI + C$$

Training!

- Data - Programs

$$N_{seq} = C_{br}Br + C_{switch}Switch + C_{FAdd}FAdd + C_{FSub}FSub + C_{Or}Or + C_{Xor}Xor + C_{Store}Store + C_{GetElementPtr}GetElementPtr + C_{FPToUI}FPToUI + C$$

Training!

- Data - Programs - Histograms

$$N_{seq} = C_{br}Br + C_{switch}Switch + C_{FAdd}FAdd + C_{FSub}FSub + C_{Or}Or + C_{Xor}Xor + C_{Store}Store + C_{GetElementPtr}GetElementPtr + C_{FPToUI}FPToUI + C$$

Training!

- Data - Programs - Histograms

Training!

- Data - Programs - Histograms

```
bool Histogram::runOnFunction(llvm::Function& function) {
  llvm::DenseMap<unsigned, unsigned> hist;
  hist.init(NUM_OPCODES);
  for (llvm::Instruction &inst: llvm::instructions(function)) {
    const unsigned op_code = inst.getOpcode();
    hist[op_code]++;
  }
}
```

Training!

- Data - Programs - Histograms
- Calibrate a Linear Model

Example: Linear Model in R

- Data - Programs - Histograms
- Calibrate a Linear Model

Example: Linear Model in R

https://www.dcc.ufmg.br/~fernando/classes/dcc888/ementa/data_ml/leaf_functions_angha.csv

Example: Linear Model in R

```
file_name = "~/Downloads/leaf_functions_angha.csv"
```

Example: Linear Model in R

```
file_name = "~/Downloads/leaf_functions_angha.csv"

data <- data[sample(1:nrow(data)), ]
```

Example: Linear Model in R

```
file_name = "~/Downloads/leaf_functions_angha.csv"

data <- data[sample(1:nrow(data)), ]

train_size <- ceiling(0.8*nrow(data))

train <- data[1:train_size,]
```

Example: Linear Model in R

```
file_name = "~/Downloads/leaf_functions_angha.csv"

data <- data[sample(1:nrow(data)), ]

train_size <- ceiling(0.8*nrow(data))

train <- data[1:train_size,]
```

Example: Linear Model in R

```
file_name = "~/Downloads/leaf_functions_angha.csv"

data <- data[sample(1:nrow(data)), ]

train_size <- ceiling(0.8*nrow(data))

train <- data[1:train_size,]

model <- lm(Os ~ Br + Switch + FAdd + FSub + Or + Xor
+ Store + GetElementPtr + FPToUI, data=train)
```

Example: Linear Model in R

```
file_name = "~/Downloads/leaf_functions_angha.csv"

data <- data[sample(1:nrow(data)), ]

train_size <- ceiling(0.8*nrow(data))

train <- data[1:train_size,]

model <- lm(Os ~ Br + Switch + FAdd + FSub + Or + Xor
+ Store + GetElementPtr + FPToUI, data=train)
```

Example: Linear Model in R

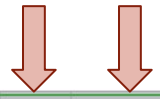
```
file_name = "~/Downloads/leaf_functions_angha.csv"

data <- data[sample(1:nrow(data)), ]

train_size <- ceiling(0.8*nrow(data))

train <- data[1:train_size,]

model <- lm(Os ~ Br + Switch + FAdd + FSub + Or + Xor
+ Store + GetElementPtr + FPToUI, data=train)
```

Example: Linear Model in R

```
file_name = "~/Downloads/leaf_functions_angha.csv"

data <- data[sample(1:nrow(data)), ]

train_size <- ceiling(0.8*nrow(data))

train <- data[1:train_size,]

model <- lm(Os ~ Br + Switch + FAdd + FSub + Or + Xor
+ Store + GetElementPtr + FPToUI, data=train)
```

Example: Linear Model in R

```
file_name = "~/Downloads/leaf_functions_angha.csv"

data <- data[sample(1:nrow(data)), ]

train_size <- ceiling(0.8*nrow(data))

train <- data[1:train_size,]

model <- lm(Os ~ Br + Switch + FAdd + FSub + Or + Xor
+ Store + GetElementPtr + FPToUI, data=train)
```



Benchmark	Zero	Ret	Br	Switch	IndirectBr	Inoke
extr_.BGAQRCode-A	0	1	145	0	0	0
extr_.BGAQRCode-A	0	1	62	0	0	0
extr_.BGAQRCode-A	0	1	126	0	0	0

...

mem2reg	O0	Os	Oz
289	366	83	83
238	327	99	99
299	383	99	99

Example: Linear Model in R

```
file_name = "~/Downloads/leaf_functions_angha.csv"

data <- data[sample(1:nrow(data)), ]

train_size <- ceiling(0.8*nrow(data))

train <- data[1:train_size,]

model <- lm(Os ~ Br + Switch + FAdd + FSub + Or + Xor
+ Store + GetElementPtr + FPToUI, data=train)
```

$$Os = C_{br}Br + C_{switch}Switch + C_{FAdd}FAdd + C_{FSub}FSub + C_{Or}Or + C_{Xor}Xor + C_{Store}Store + C_{GetElementPtr}GetElementPtr + C_{FPToUI}FPToUI + C$$

Example: Linear Model in R

```
file_name = "~/Downloads/leaf_functions_angha.csv"

data <- data[sample(1:nrow(data)), ]

train_size <- ceiling(0.8*nrow(data))

train <- data[1:train_size,]

model <- lm(Os ~ Br + Switch + FAdd + FSub + Or + Xor
+ Store + GetElementPtr + FPToUI, data=train)

test <- data[(train_size+1):nrow(data),]

results <- predict(model, newdata = test)
```

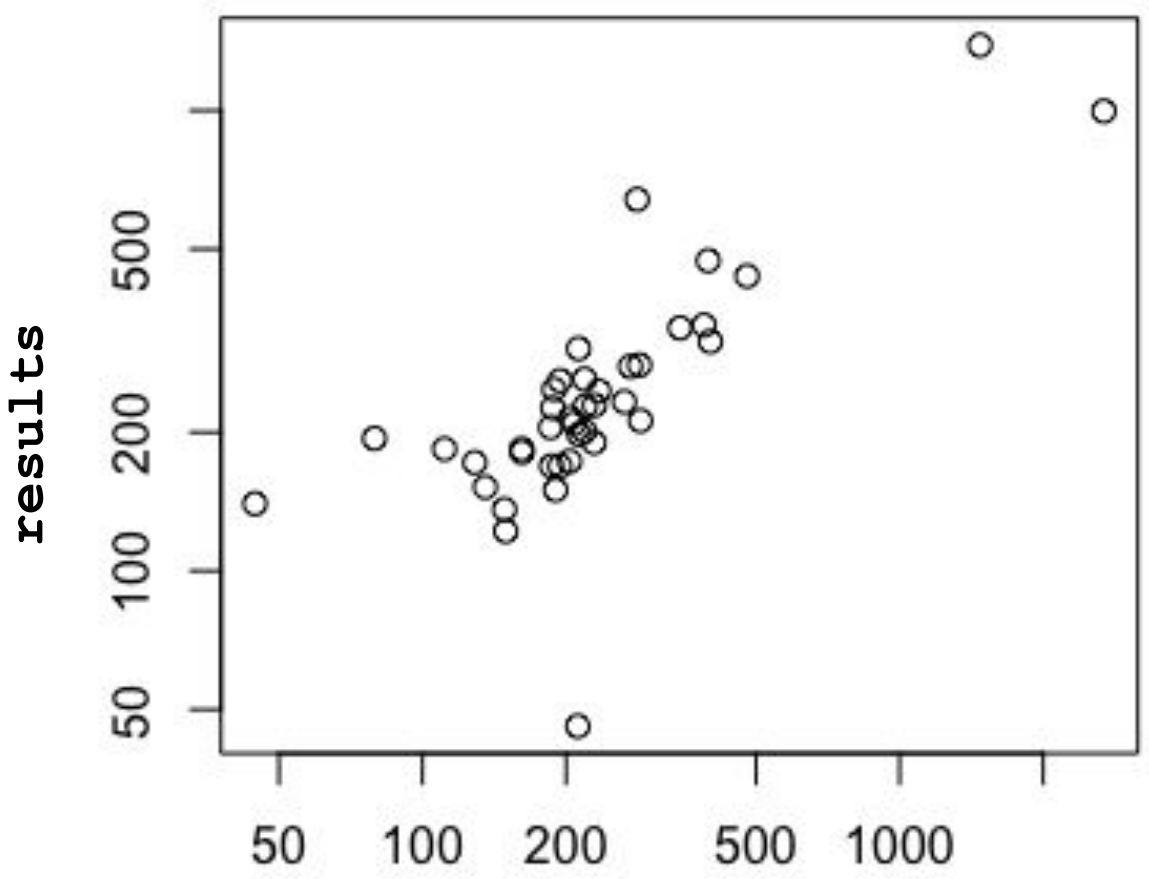
Example: Linear Model in R

```
file_name = "~/Downloads/leaf_functions_angha.csv"

data <- data[sample(1:nrow(data)), ]

train_size <- ceiling(0.8*nrow(data))

train <- data[1:train_size,]

model <- lm(Os ~ Br + Switch + FAdd + FSub + Or + Xor
+ Store + GetElementPtr + FPToUI, data=train)

test <- data[(train_size+1):nrow(data),]

results <- predict(model, newdata = test)
```

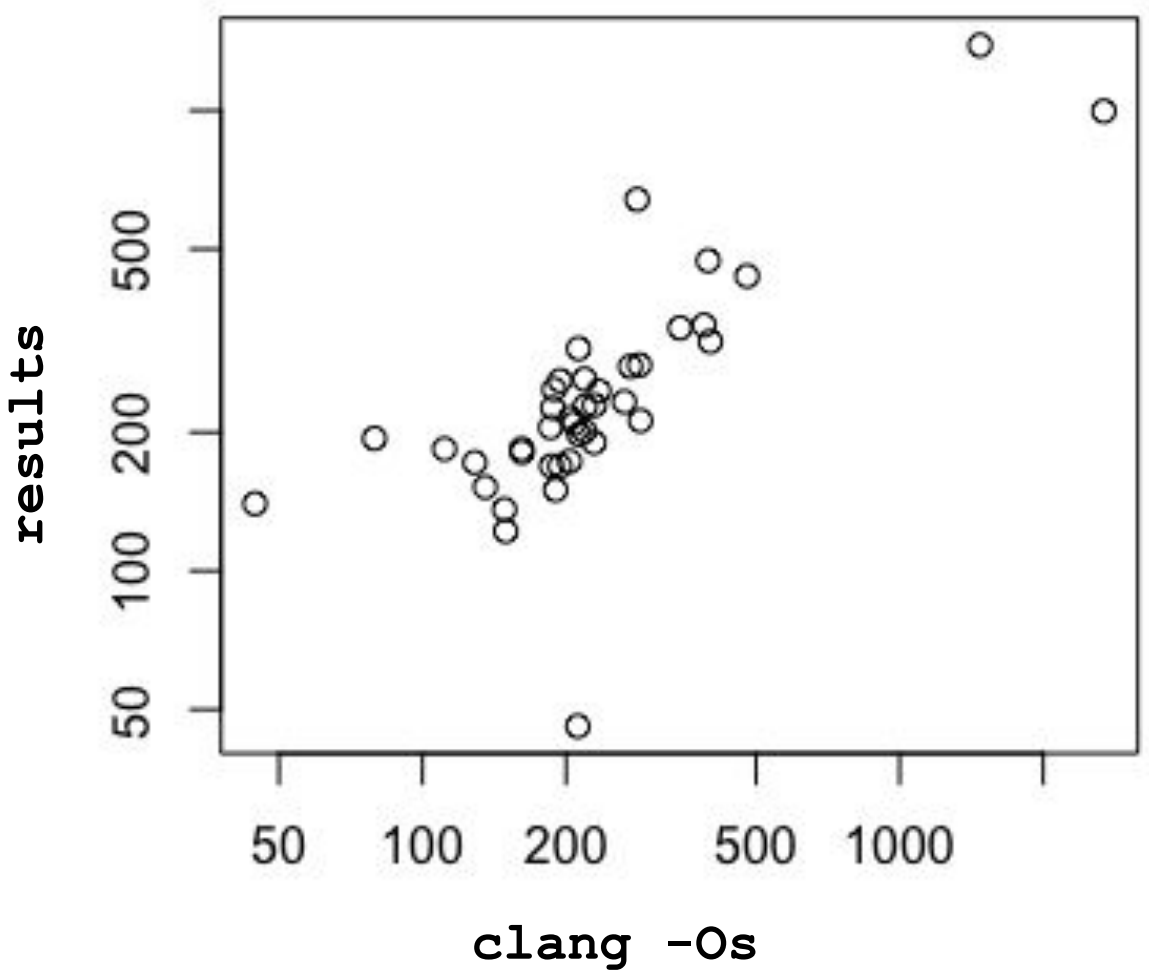
Example: Linear Model in R

```
file_name = "~/Downloads/leaf_functions_angha.csv"

data <- data[sample(1:nrow(data)), ]

train_size <- ceiling(0.8*nrow(data))

train <- data[1:train_size,]

model <- lm(Os ~ Br + Switch + FAdd + FSub + Or + Xor
+ Store + GetElementPtr + FPToUI, data=train)

test <- data[(train_size+1):nrow(data),]

results <- predict(model, newdata = test)

plot(test$Os ~ results, log = 'xy')
```

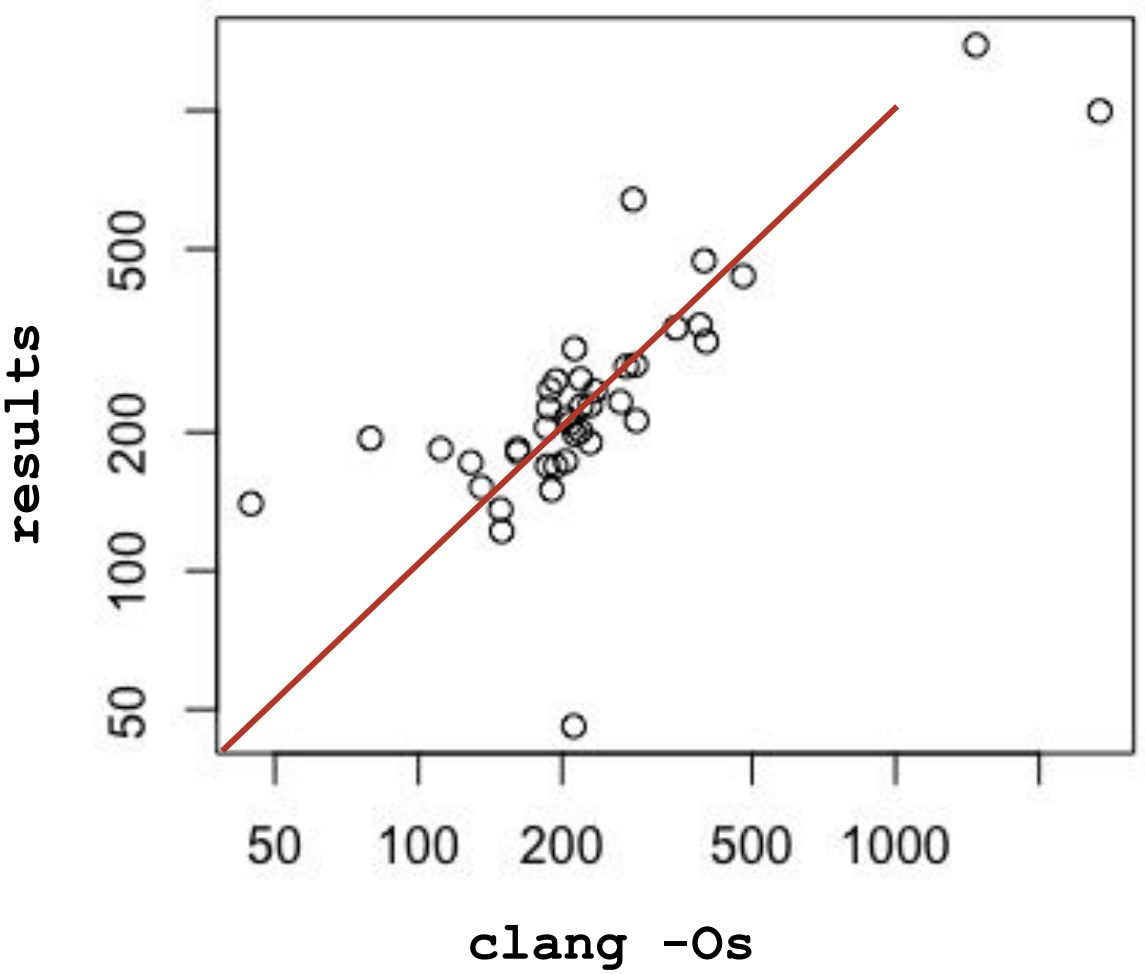
Example: Linear Model in R



Example: Linear Model in R



Example: Linear Model in R



Understanding the Model

```
model <- lm(Os ~ Br + Switch + FAdd + FSub + Or + Xor
+ Store + GetElementPtr + FPToUI, data=train)
```

Understanding the Model

```
model <- lm(Os ~ Br + Switch + FAdd + FSub + Or + Xor
+ Store + GetElementPtr + FPToUI, data=train)
```

$$N_{seq} = C_{br}Br + C_{switch}Switch + C_{FAdd}FAdd + C_{FSub}FSub + C_{Or}Or + C_{Xor}Xor + C_{Store}Store + C_{GetElementPtr}GetElementPtr + C_{FPToUI}FPToUI + C$$

Understanding the Model

```
model <- lm(Os ~ Br + Switch + FAdd + FSub + Or + Xor
+ Store + GetElementPtr + FPToUI, data=train)

summary(model)
```

$$N_{seq} = C_{br}Br + C_{switch}Switch + C_{FAdd}FAdd + C_{FSub}FSub + C_{Or}Or + C_{Xor}Xor + C_{Store}Store + C_{GetElementPtr}GetElementPtr + C_{FPToUI}FPToUI + C$$

Understanding the Model

```
model <- lm(Os ~ Br + Switch + FAdd + FSub + Or + Xor
+ Store + GetElementPtr + FPToUI, data=train)
```

```
summary(model)
```

Coefficients:

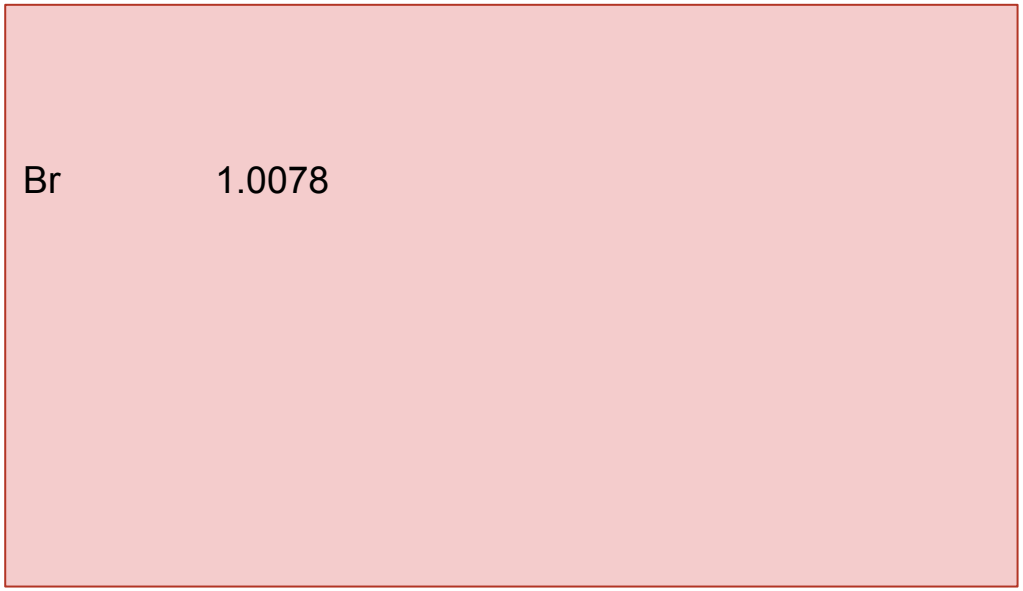
	Estimate	Std. Error	t value	Pr(>|t|)	
(Intercept)	-0.8488	12.4278	-0.068	0.945638	
Br	1.0078	0.1318	7.645	2.10e-12	***
Switch	-13.5785	6.6421	-2.044	0.042627	*
FAdd	2.0047	0.3191	6.283	3.25e-09	***
FSub	-0.4547	0.4290	-1.060	0.290803	
Or	2.1505	0.5011	4.292	3.12e-05	***
Xor	1.1496	1.2677	0.907	0.365881	
Store	1.1631	0.1323	8.794	2.71e-15	***
GetElementPtr	2.0685	0.1940	10.663	< 2e-16	***
FPToUI	0.8268	0.2223	3.719	0.000279	***

$$N_{seq} = C_{br}Br + C_{switch}Switch + C_{FAdd}FAdd + C_{FSub}FSub + C_{Or}Or + C_{Xor}Xor + C_{Store}Store + C_{GetElementPtr}GetElementPtr + C_{FPToUI}FPToUI + C$$

Understanding the Model

```
model <- lm(Os ~ Br + Switch + FAdd + FSub + Or + Xor
+ Store + GetElementPtr + FPToUI, data=train)
```

```
summary(model)
```

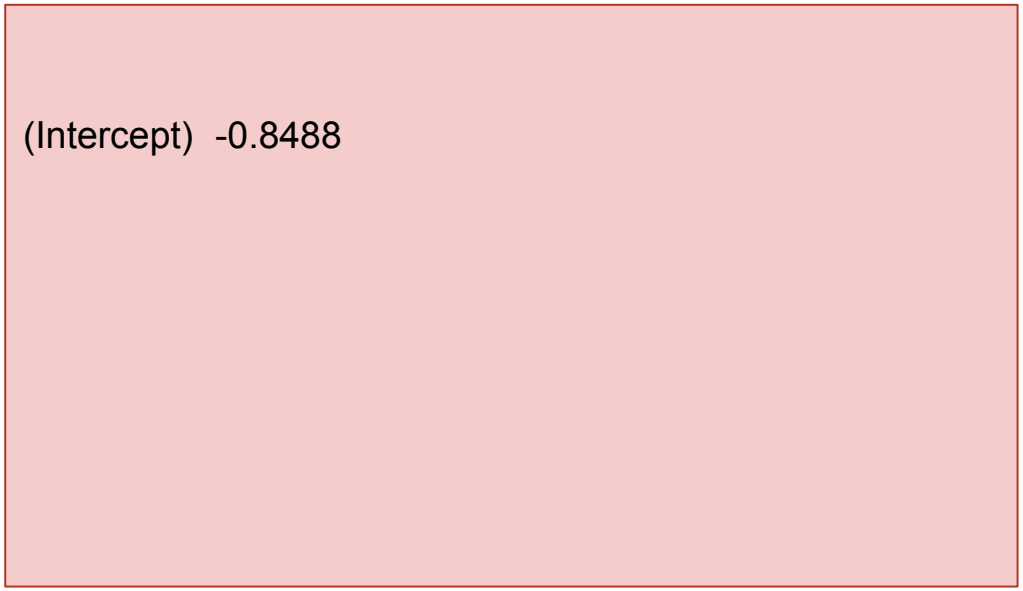


$$N_{seq} = C_{br}Br + C_{switch}Switch + C_{FAdd}FAdd + C_{FSub}FSub + C_{Or}Or + C_{Xor}Xor + C_{Store}Store + C_{GetElementPtr}GetElementPtr + C_{FPToUI}FPToUI + C$$

Understanding the Model

```
model <- lm(Os ~ Br + Switch + FAdd + FSub + Or + Xor
+ Store + GetElementPtr + FPToUI, data=train)
```

```
summary(model)
```



$$N_{seq} = C_{br}Br + C_{switch}Switch + C_{FAdd}FAdd + C_{FSub}FSub + C_{Or}Or + C_{Xor}Xor + C_{Store}Store + C_{GetElementPtr}GetElementPtr + C_{FPToUI}FPToUI + C$$

Understanding the Model

```
model <- lm(Os ~ Br + Switch + FAdd + FSub + Or + Xor
+ Store + GetElementPtr + FPToUI, data=train)
```

```
summary(model)
```

Coefficients:

	Estimate	Std. Error	t value	Pr(>|t|)	
(Intercept)	-0.8488	12.4278	-0.068	0.945638	
Br	1.0078	0.1318	7.645	2.10e-12	***
Switch	-13.5785	6.6421	-2.044	0.042627	*
FAdd	2.0047	0.3191	6.283	3.25e-09	***
FSub	-0.4547	0.4290	-1.060	0.290803	
Or	2.1505	0.5011	4.292	3.12e-05	***
Xor	1.1496	1.2677	0.907	0.365881	
Store	1.1631	0.1323	8.794	2.71e-15	***
GetElementPtr	2.0685	0.1940	10.663	< 2e-16	***
FPToUI	0.8268	0.2223	3.719	0.000279	***

$$N_{seq} = 1.0078Br - 13.5785Switch + 2.0047FAdd - 0.4547FSub + 2.1505Or + 1.1496Xor + 1.1631Store + 2.0685GetElementPtr + 0.8268FPToUI - 0.8488$$

Using the Model

```
model <- lm(Os ~ Br + Switch + FAdd + FSub + Or + Xor
+ Store + GetElementPtr + FPToUI, data=train)

summary(model)
```

```
int main(int argc, char** argv) {
  int sum = 0;
  if (argc > 1) {
   for (int i = 1; i < argc; i++) {
   sum += (int)argv[i][0];
   }
  }
  return sum;
}
```

Using the Model

```
model <- lm(Os ~ Br + Switch + FAdd + FSub + Or + Xor
+ Store + GetElementPtr + FPToUI, data=train)

summary(model)
```

```
0, 1, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 2, 0,
0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 2, 0, 3, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0
```

Using the Model

```
model <- lm(Os ~ Br + Switch + FAdd + FSub + Or + Xor
+ Store + GetElementPtr + FPToUI, data=train)

summary(model)

hf <- data.frame(Br = c(6), Switch = c(0), FAdd =
c(2), FSub = c(0), Or = c(0), Xor = c(0), Store =
c(0), GetElementPtr = c(2), FPToUI = c(0))


0, 1, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 2, 0,
0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 2, 0, 3, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0
```

Using the Model

```
model <- lm(Os ~ Br + Switch + FAdd + FSub + Or + Xor
+ Store + GetElementPtr + FPToUI, data=train)

summary(model)

hf <- data.frame(Br = c(6), Switch = c(0), FAdd =
c(2), FSub = c(0), Or = c(0), Xor = c(0), Store =
c(0), GetElementPtr = c(2), FPToUI = c(0))
```

```
predict(model, newdata = hf)
13.34451
```

Using the Model

```
coefs <- c(1.0078, -13.5785, 2.0047, -0.4547, 2.1505,
1.1496, 1.1631, 2.0685, 0.8268)
```

```
predict(model, newdata = hf)
13.34451
```

Using the Model

```
coefs <- c(1.0078, -13.5785, 2.0047, -0.4547, 2.1505,
1.1496, 1.1631, 2.0685, 0.8268)

histg <- c(6, 0, 2, 0, 0, 0, 0, 2, 0)
```

```
predict(model, newdata = hf)
13.34451
```

Using the Model

```
coefs <- c(1.0078, -13.5785, 2.0047, -0.4547, 2.1505,
1.1496, 1.1631, 2.0685, 0.8268)

histg <- c(6, 0, 2, 0, 0, 0, 0, 2, 0)

coefs %*% histg - 0.8488
```

```
         [,1]
[1,] 13.3444
```

```
predict(model, newdata = hf)
13.34451
```

Using the Model

```
model <- lm(Os ~ Br + Switch + FAdd + FSub + Or + Xor
+ Store + GetElementPtr + FPToUI, data=train)

summary(model)

hf <- data.frame(Br = c(6), Switch = c(0), FAdd =
c(2), FSub = c(0), Or = c(0), Xor = c(0), Store =
c(0), GetElementPtr = c(2), FPToUI = c(0))
```

```
predict(model, newdata = hf)
13.34451
```

How would you check the number of instructions in the actual program, optimized with clang -Os?

Using the Model

```
clang -S -emit-llvm -Os file.c -o file.opt.ll

opt -instcount -stats -disable-output file.opt.ll
```

```
predict(model, newdata = hf)
13.34451
```

How would you check the number of instructions in the actual program, optimized with clang -Os?

Using the Model

```
clang -S -emit-llvm -Os file.c -o file.opt.ll

opt -instcount -stats -disable-output file.opt.ll
```

```
 2 instcount - Number of Add insts
 3 instcount - Number of Br insts
 ...
 1 instcount - Number of non-external functions
16 instcount - Number of instructions (of all types)
```

```
predict(model, newdata = hf)
13.34451
```

How would you check the number of instructions in the actual program, optimized with clang -Os?

Using the Model

```
clang -S -emit-llvm -O0 file.c -o file.opt.ll

opt -instcount -stats -disable-output file.opt.ll
```

```
 2 instcount - Number of Add insts
 5 instcount - Number of Br insts
 ...
 1 instcount - Number of non-external functions
37 instcount - Number of instructions (of all types)
```

```
predict(model, newdata = hf)
13.34451
```

Other Examples

Other Examples

Fernando Magno Quintão Pereira, Guilherme V. Leobas, Abdoulaye Gamatié:
Static Prediction of Silent Stores.
ACM Trans. Archit. Code Optim. 15(4): 44:1-44:26 (2019)

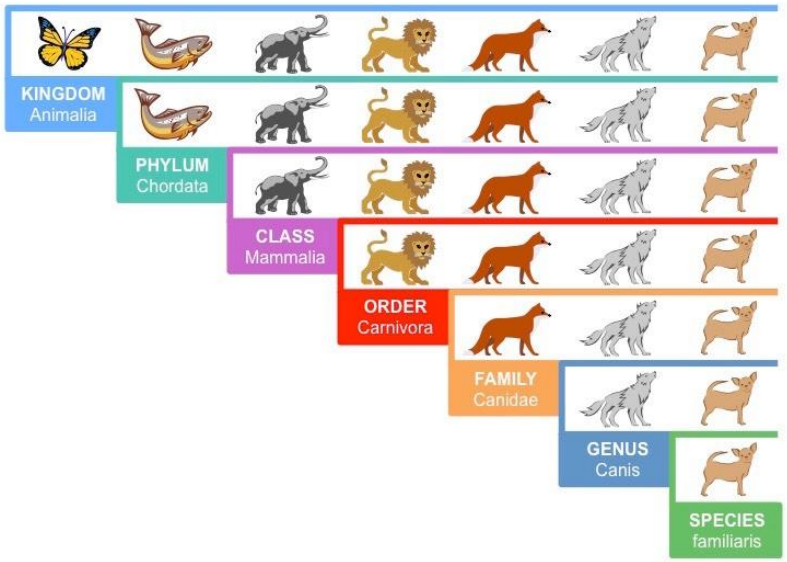
Other Examples

Junio R. da Silva, Lorena Leão, Vinicius Petrucci, Abdoulaye Gamatié, Fernando M. Q. Pereira: *Mapping Computations in Heterogeneous Multicore Systems with Statistical Regression on Program Inputs.* ACM Trans. Embed. Comput. Syst. 20(6): 112:1-112:35 (2021)

Fernando Magno Quintão Pereira, Guilherme V. Leobas, Abdoulaye Gamatié: *Static Prediction of Silent Stores.* ACM Trans. Archit. Code Optim. 15(4): 44:1-44:26 (2019)

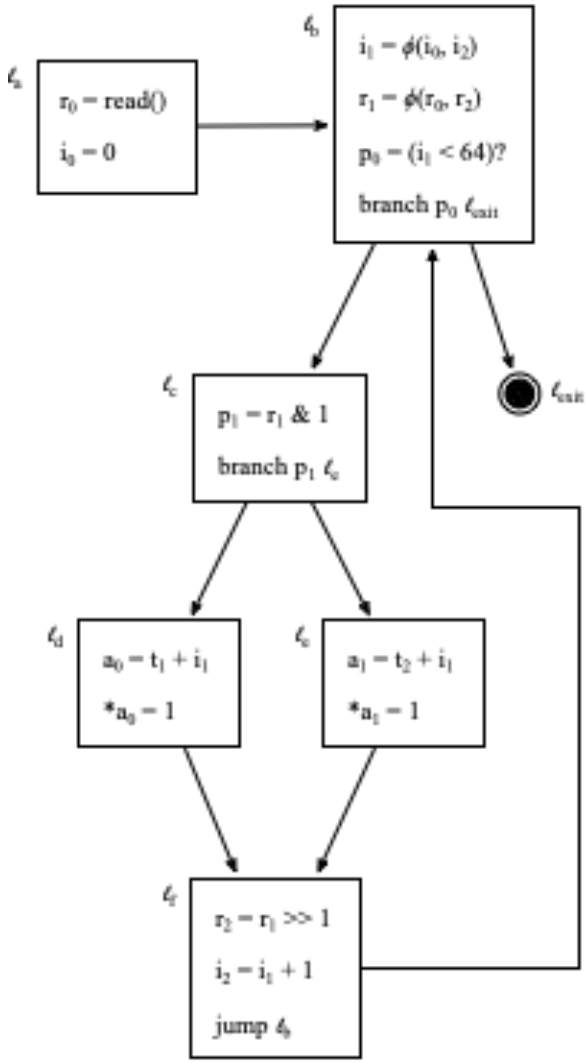
Compilers Laboratory

Classification

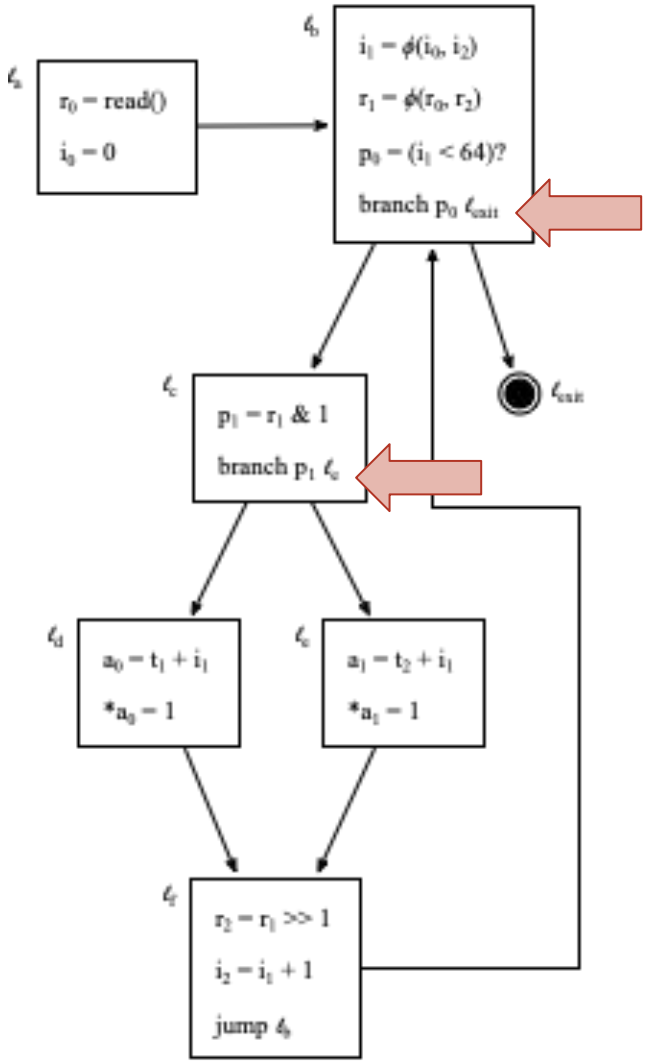


Branches

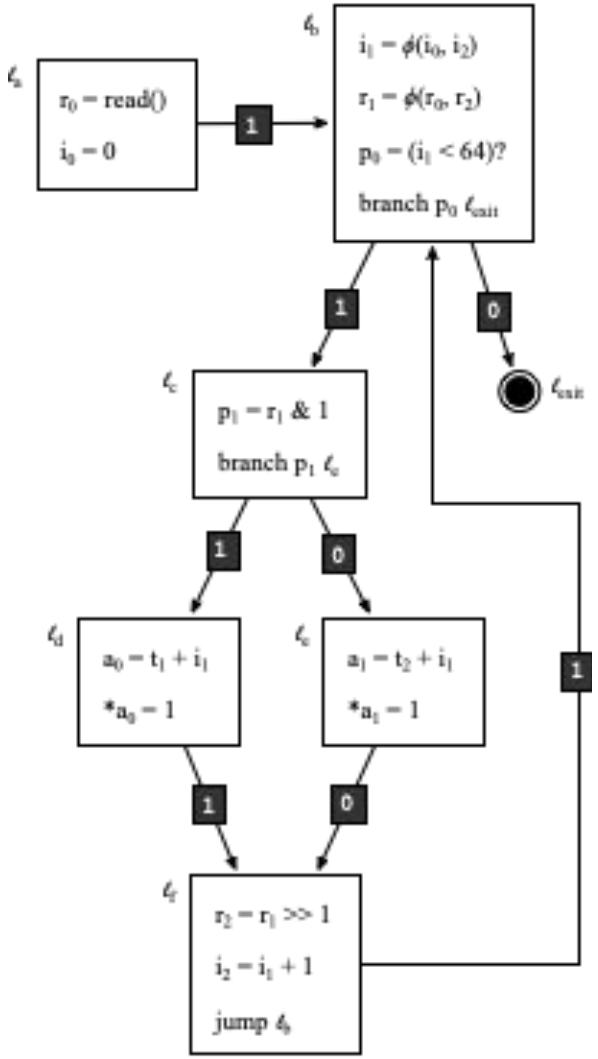
Branches


ℓ_a
$r_0 = \text{read}()$
$i_0 = 0$
ℓ_b
$i_1 = \phi(i_0, i_2)$
$r_1 = \phi(r_0, r_2)$
$p_0 = (i_1 < 64)?$
branch p_0 ℓ_{exit}
ℓ_c
$p_1 = r_1 \,\&\, 1$
branch p_1 ℓ_e
ℓ_{exit}
ℓ_d
$a_0 = t_1 + i_1$
$*a_0 = 1$
ℓ_e
$a_1 = t_2 + i_1$
$*a_1 = 1$
ℓ_f
$r_2 = r_1 >> 1$
$i_2 = i_1 + 1$
jump ℓ_b

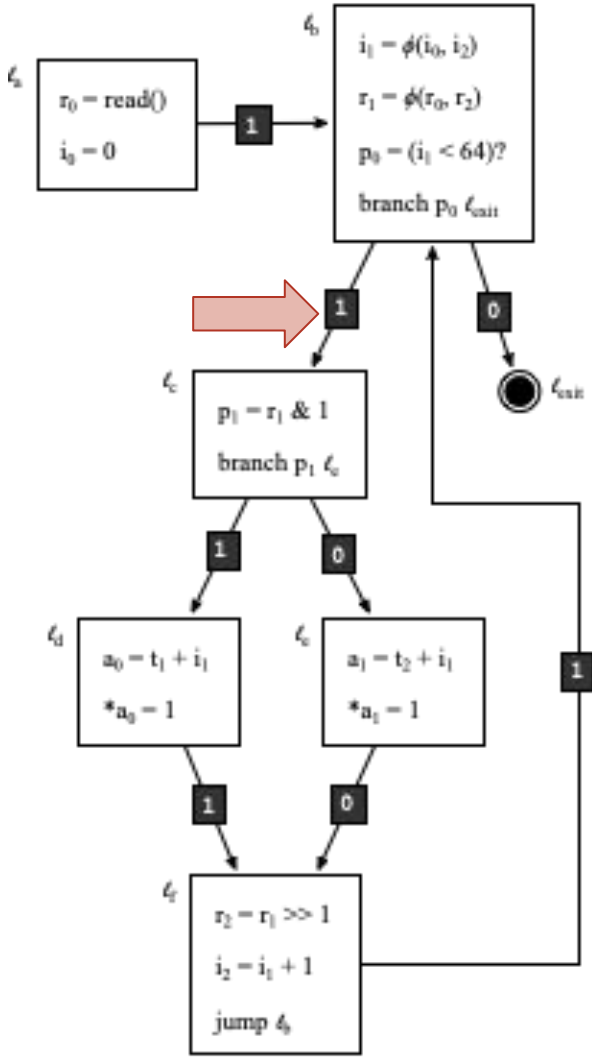
Branches


ℓ_a
$r_0 = \text{read}()$
$i_0 = 0$
ℓ_b
$i_1 = \phi(i_0, i_2)$
$r_1 = \phi(r_0, r_2)$
$p_0 = (i_1 < 64)?$
branch p_0 ℓ_{exit}
ℓ_c
$p_1 = r_1 \,\&\, 1$
branch p_1 ℓ_e
ℓ_{exit}
ℓ_d
$a_0 = t_1 + i_1$
$*a_0 = 1$
ℓ_e
$a_1 = t_2 + i_1$
$*a_1 = 1$
ℓ_f
$r_2 = r_1 >> 1$
$i_2 = i_1 + 1$
jump ℓ_b

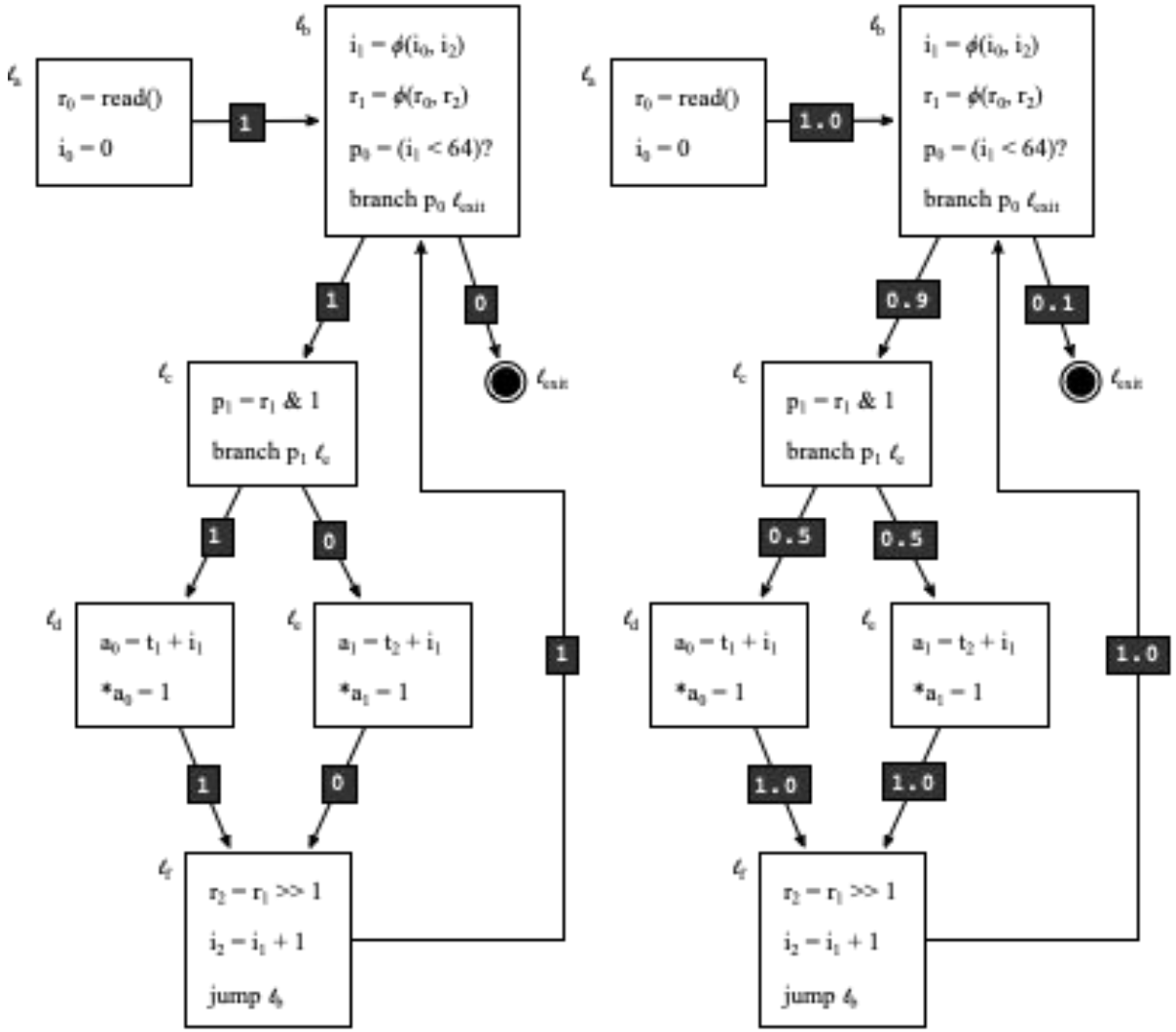
The Classification Problem



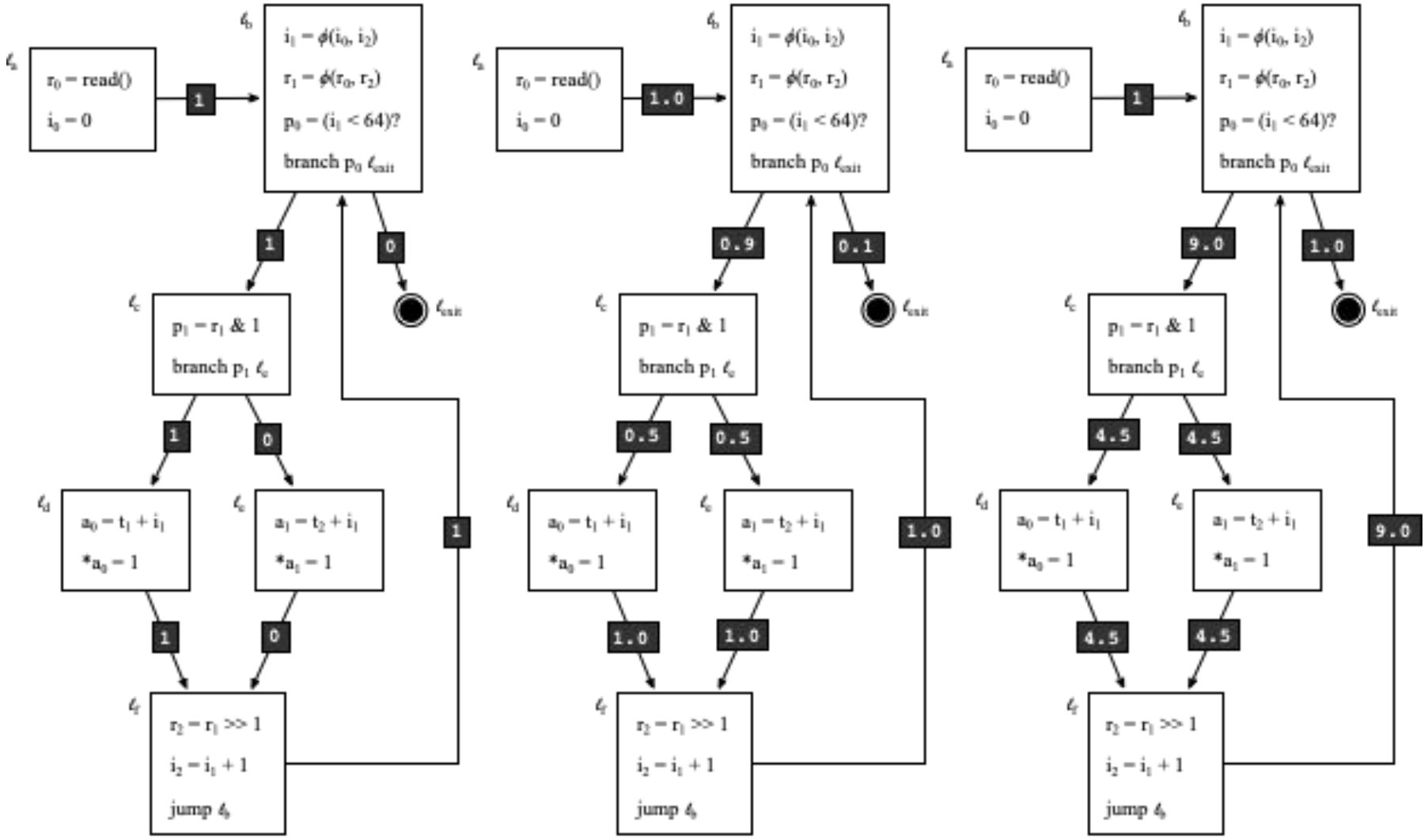
The Classification Problem



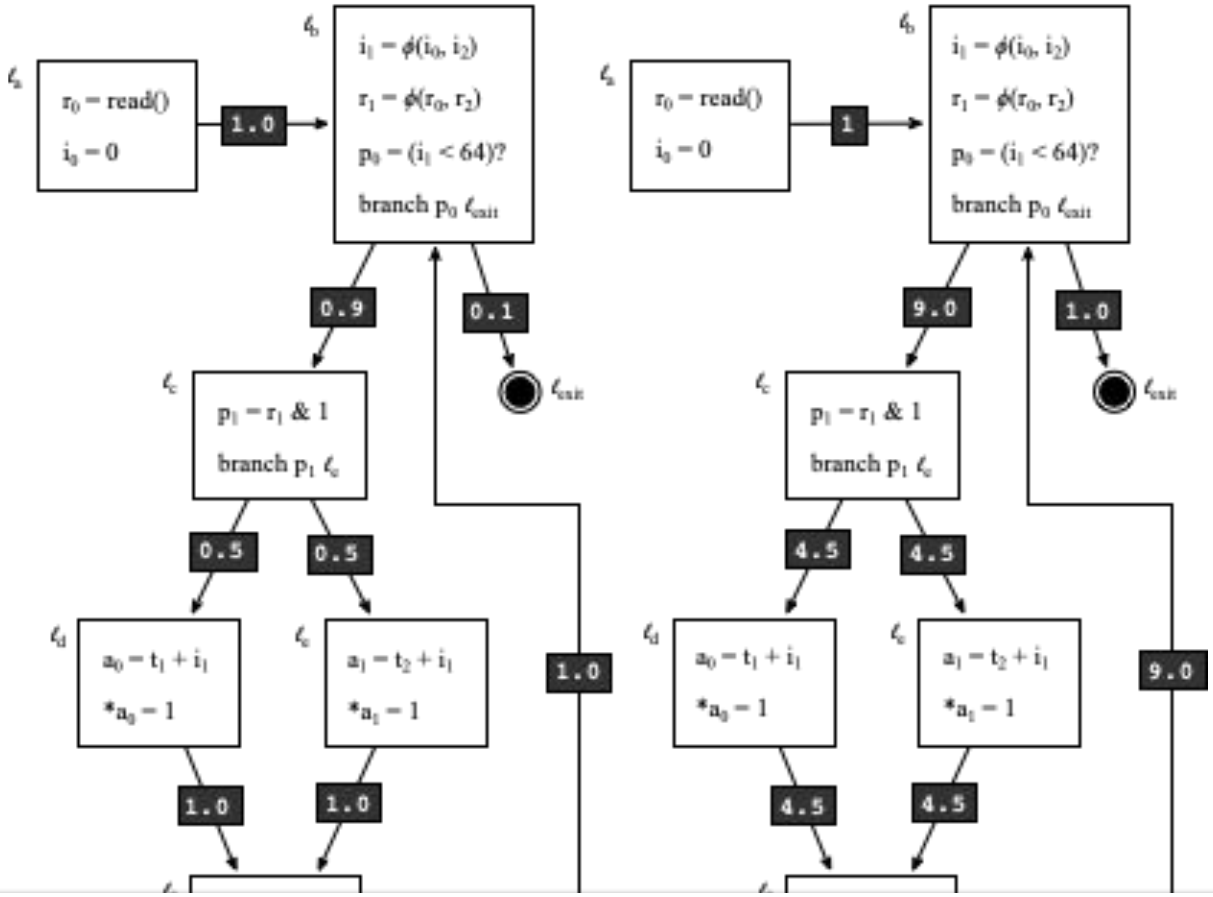
Probabilities



Regression Problems

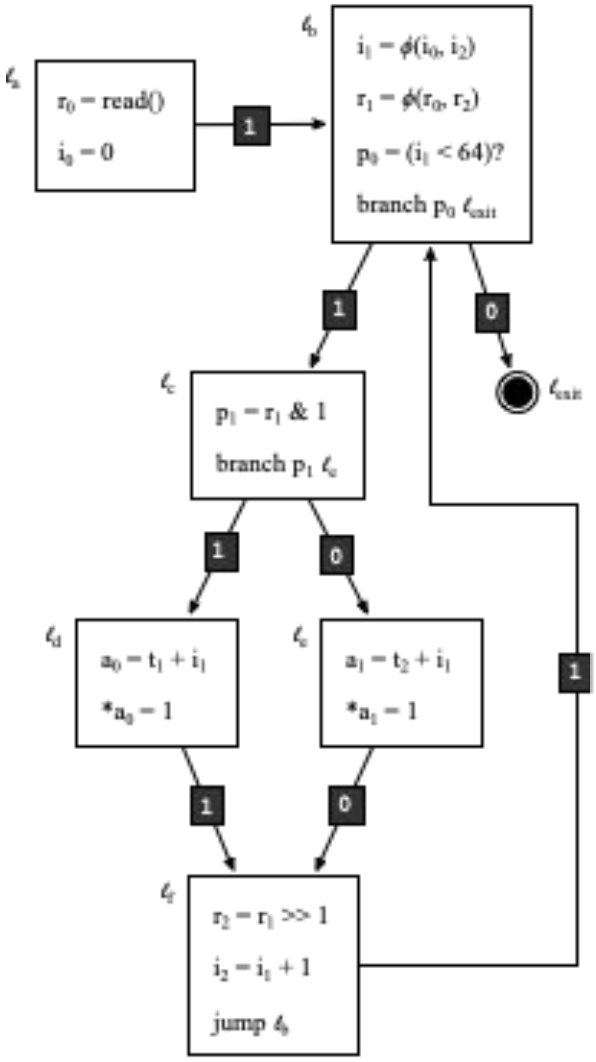


Regression Problems

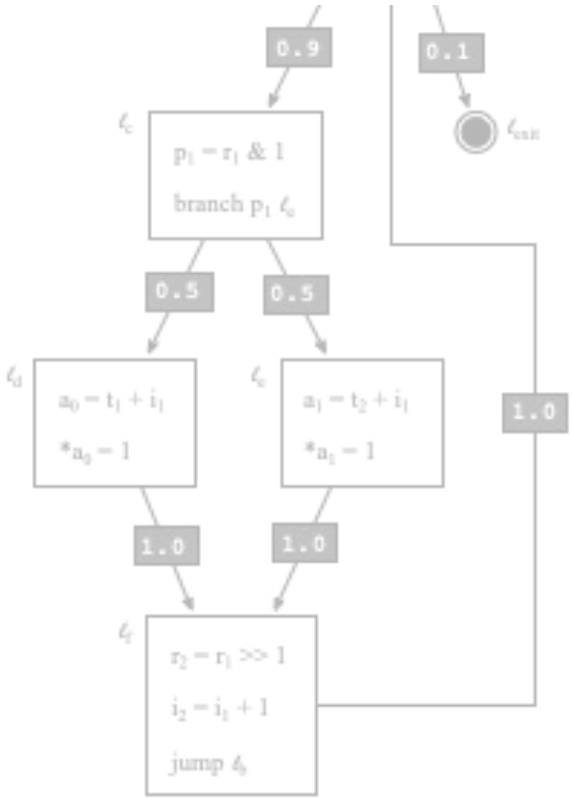


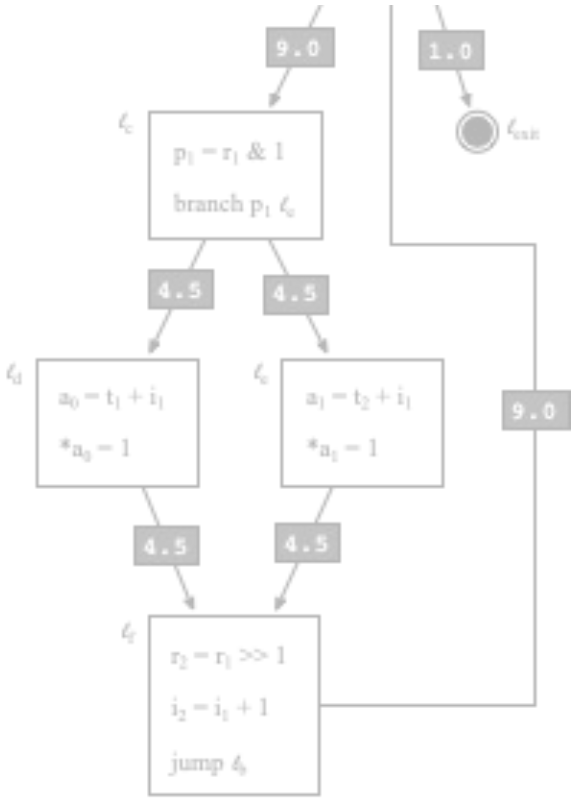
Angélica Aparecida Moreira, Guilherme Ottoni, Fernando Magno Quintão Pereira: *VESPA: static profiling for binary optimization.* Proc. ACM Program. Lang. 5(OOPSLA): 1-28 (2021)

Predicting Branches

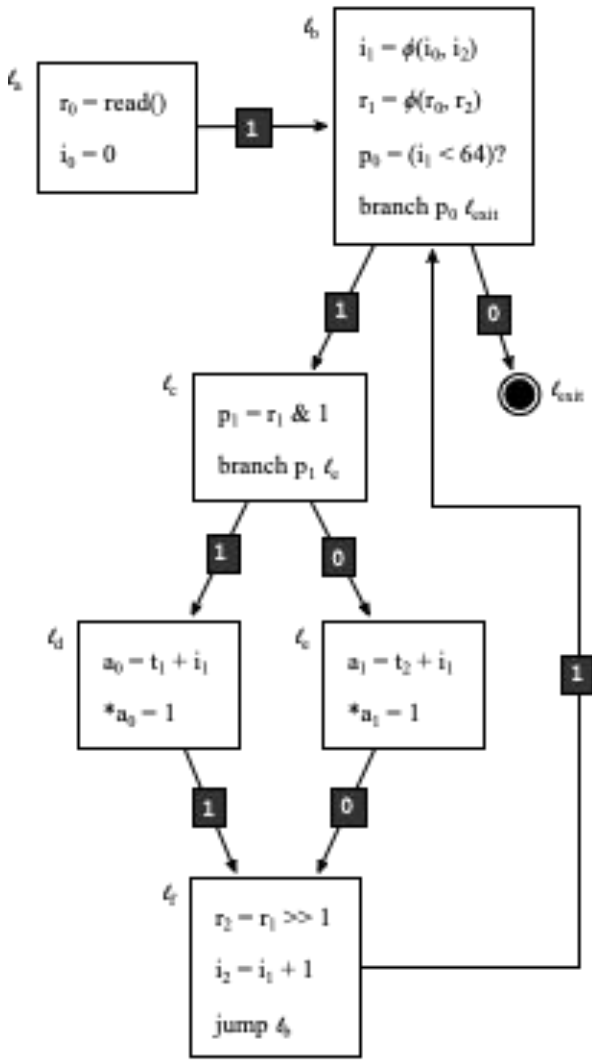


Brad Calder, Dirk Grunwald, M. Jones, Donald Lindsay, James Martin, Michael Mozer, B. Zorn:
Evidence-Based Static Branch Prediction Using Machine Learning.
ACM Trans. Program. Lang. Syst. 19(1): 188-222 (1997)





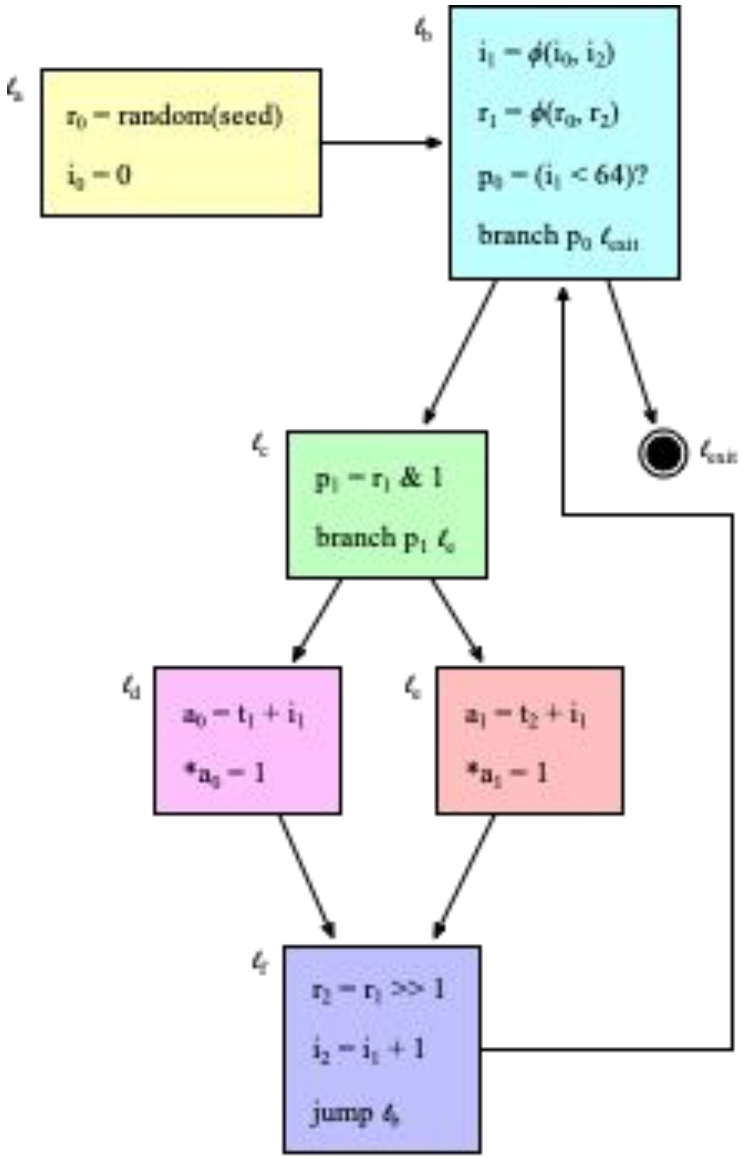
Some Motivation



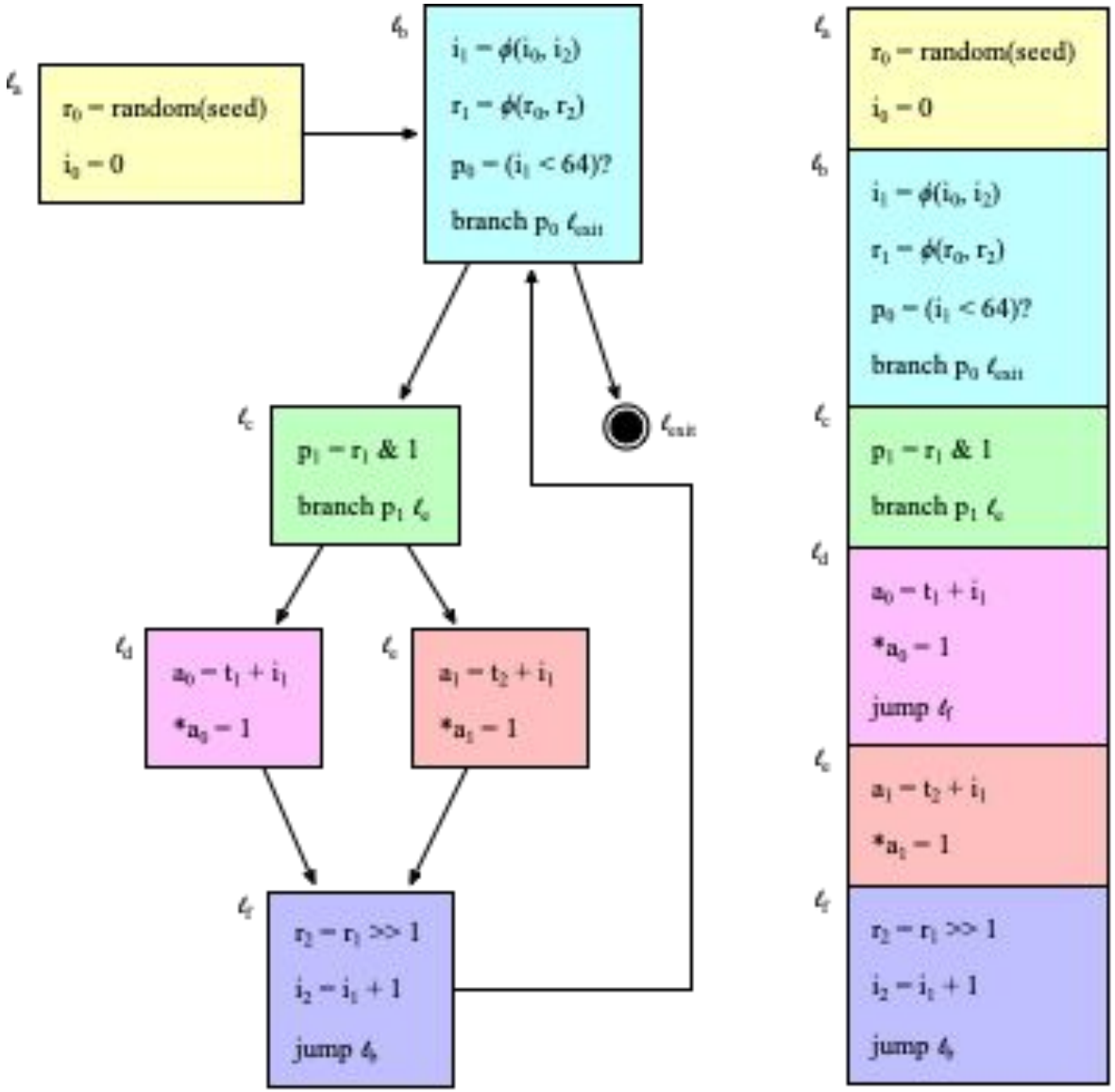
Brad Calder, Dirk Grunwald, M. Jones, Donald Lindsay, James Martin, Michael Mozer, B. Zorn:
Evidence-Based Static Branch Prediction Using Machine Learning.
ACM Trans. Program. Lang. Syst. 19(1): 188-222 (1997)

What can you do with this information?

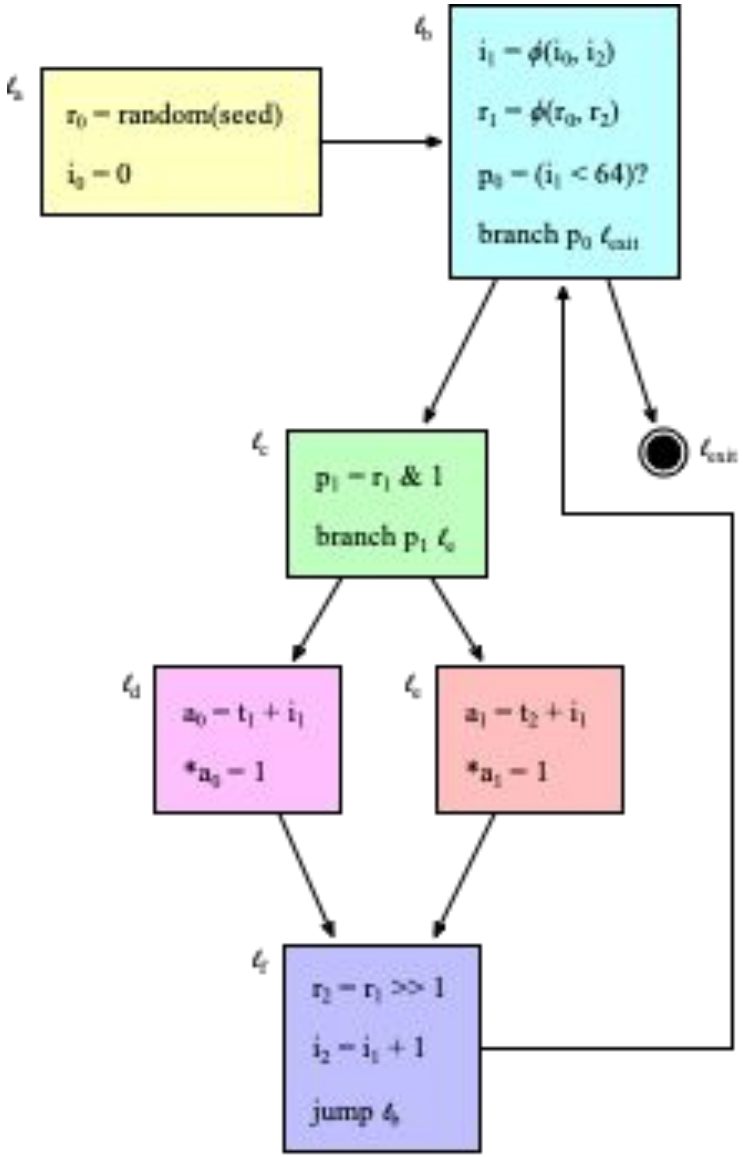
Some Motivation

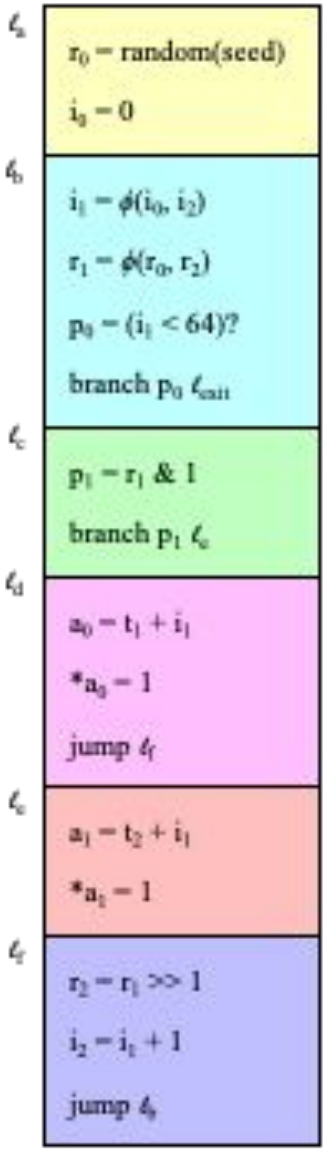


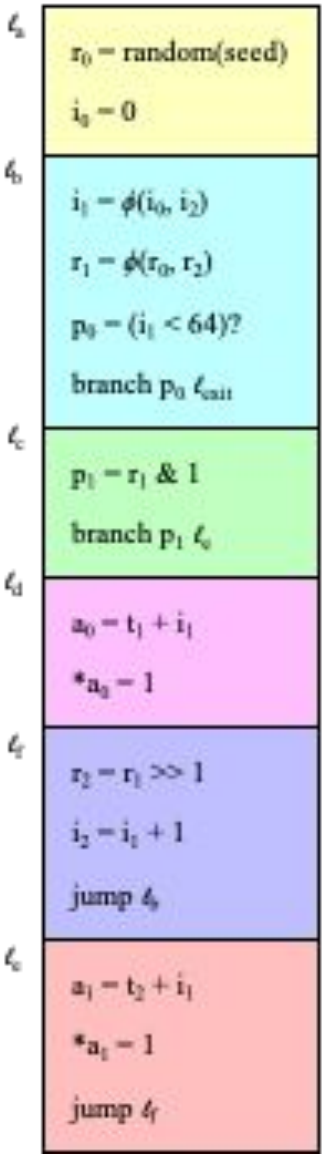
Some Motivation



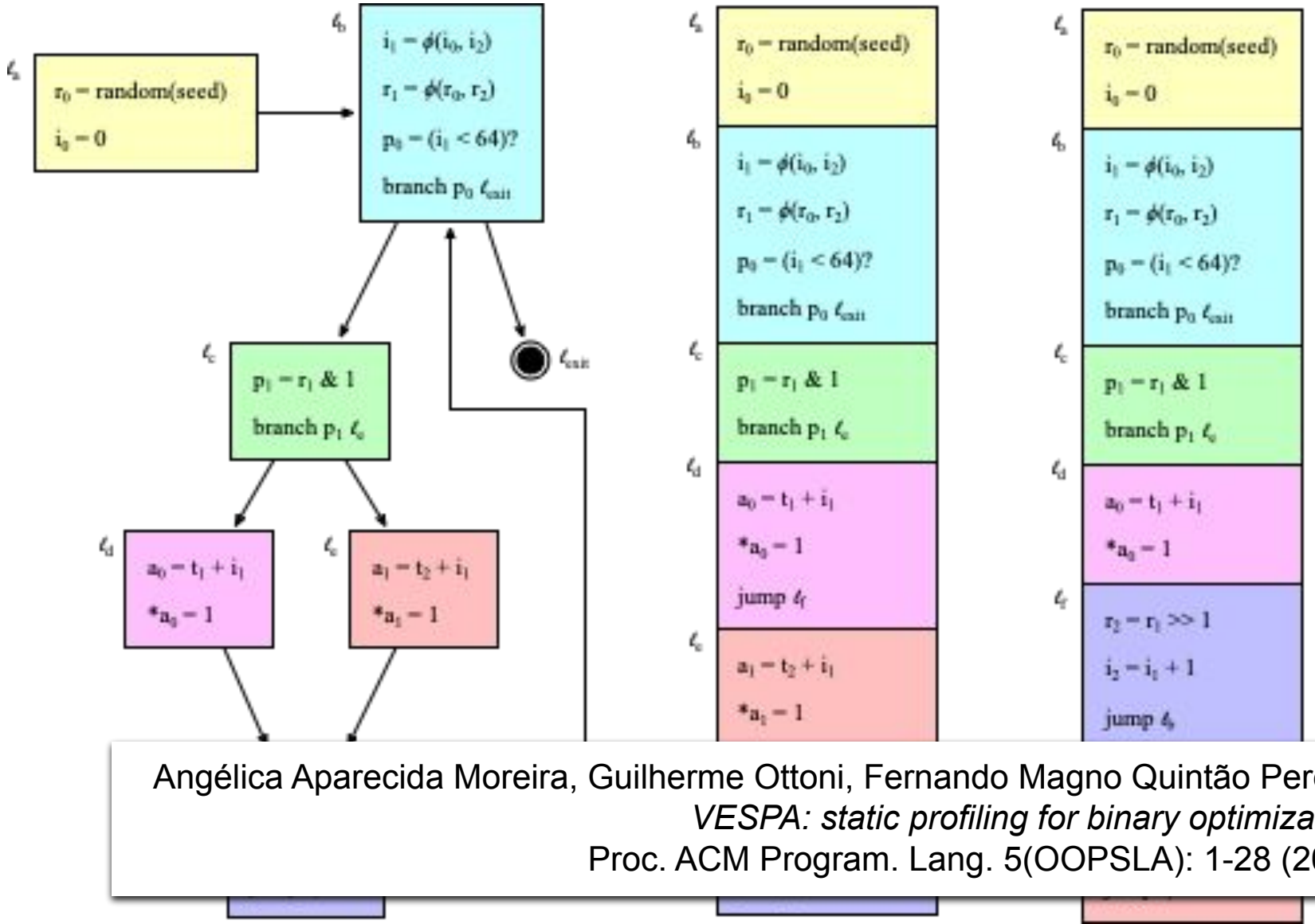
Some Motivation







Some Motivation



Angélica Aparecida Moreira, Guilherme Ottoni, Fernando Magno Quintão Pereira:
VESPA: static profiling for binary optimization.
Proc. ACM Program. Lang. 5(OOPSLA): 1-28 (2021)

Logistic Regression

Logistic Regression

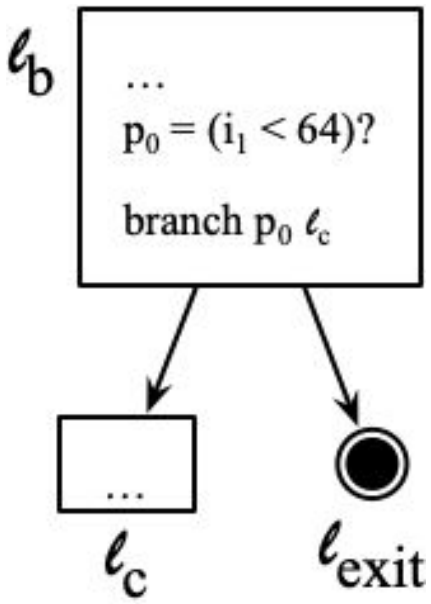
- Linear Regression

Logistic Regression

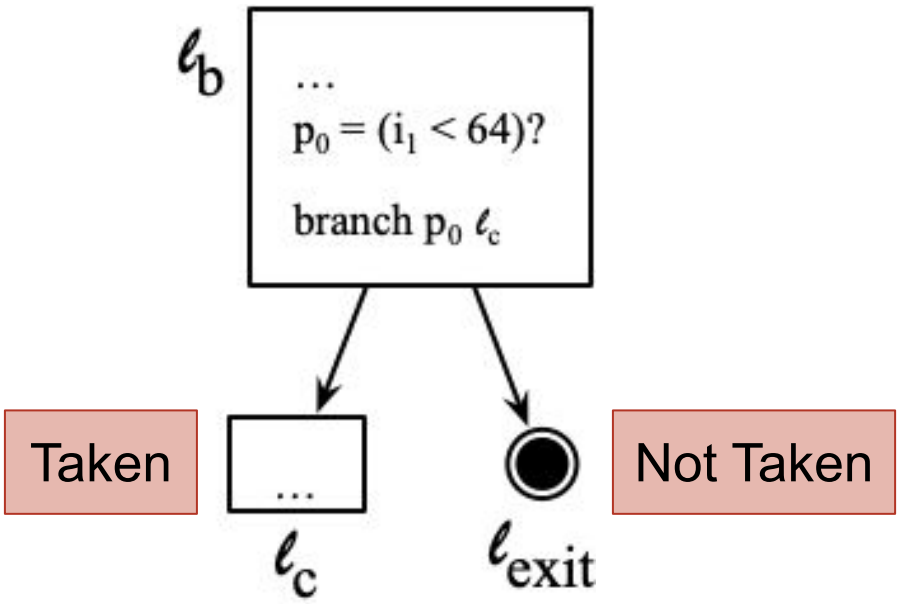
- ~~Linear Regression~~ Linear Classification

Branch Classification

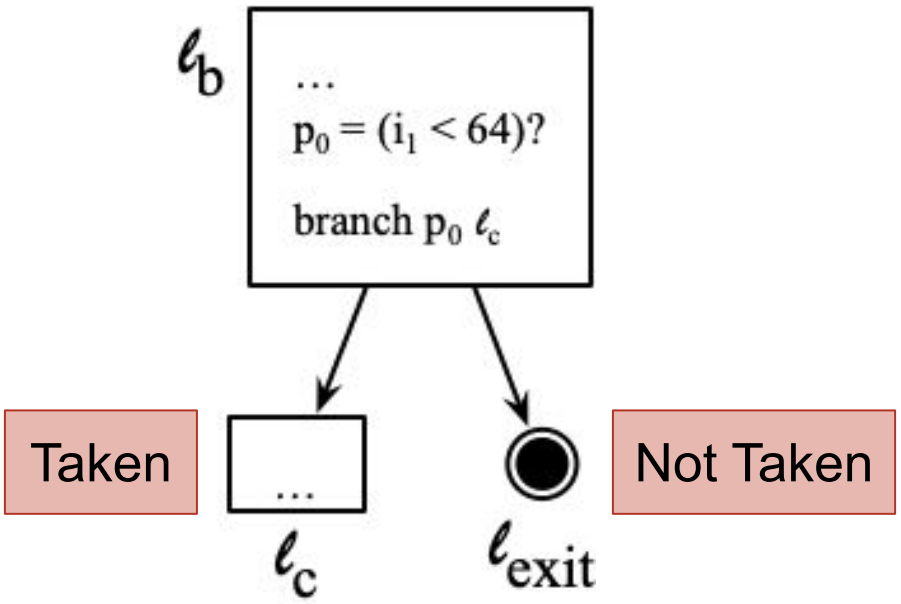
Branch Classification



Branch Classification

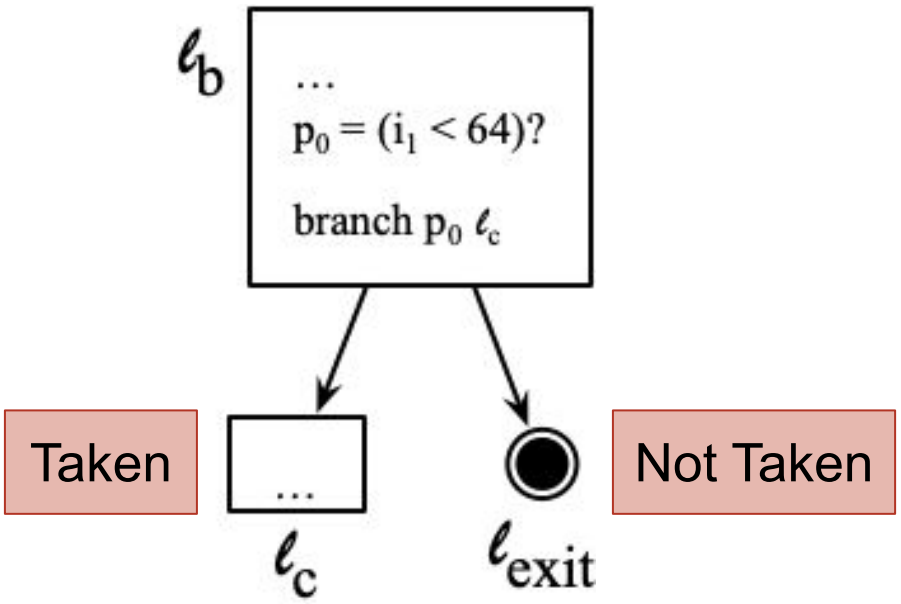


Branch Classification



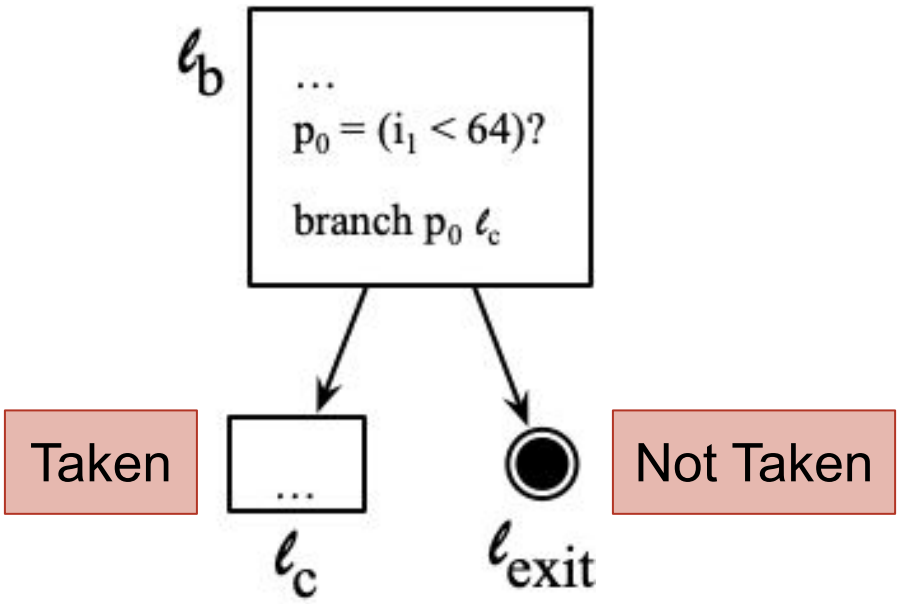
$$P_{br} = C_{Break}Break + C_{BackEdge}BackEdge + C_{EqZero}EqZero + C_{IsZero}IsZero + C_{NoPDom}NoPDom + C_{ToFun}ToFun + C$$

Branch Classification



$$P_{br} = C_{Break}\,Break + C_{BackEdge}\,BackEdge + C_{EqZero}\,EqZero + C_{IsZero}\,IsZero + C_{NoPDom}\,NoPDom + C_{ToFun}\,ToFun + C$$

The Logistic Function



$$P_{br} = \sigma(C_{Break}Break + C_{BackEdge}BackEdge + C_{EqZero}EqZero + C_{IsZero}IsZero + C_{NoPDom}NoPDom + C_{ToFun}ToFun + C)$$

The Logistic Function

$$\sigma(x) = \frac{1}{1 + e^{-(x)}}$$

$$P_{br} = \sigma(C_{Break}Break + C_{BackEdge}BackEdge + C_{EqZero}EqZero + C_{IsZero}IsZero + C_{NoPDom}NoPDom + C_{ToFun}ToFun + C)$$

The Logistic Function

$$\sigma(x) = \frac{1}{1 + e^{-(x)}}$$

```
sig <- function(x) {
  1/(1 + exp(-x))
}
```

The Logistic Function

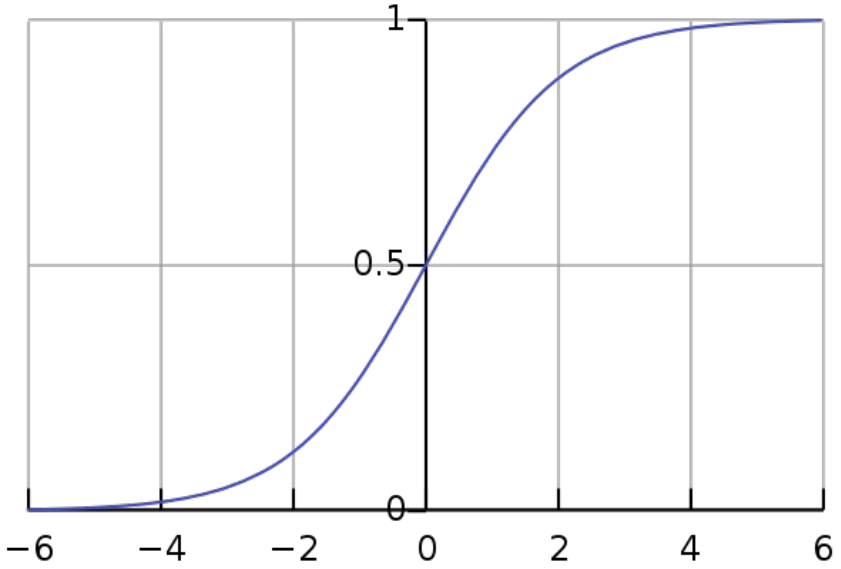
$$\sigma(x) = \frac{1}{1+e^{-(x)}}$$

```
sig <- function(x) {
  1/(1 + exp(-x))
}

plot(sigmoid, -6, 6)
```

The Logistic Function

$$\sigma(x) = \frac{1}{1 + e^{-(x)}}$$

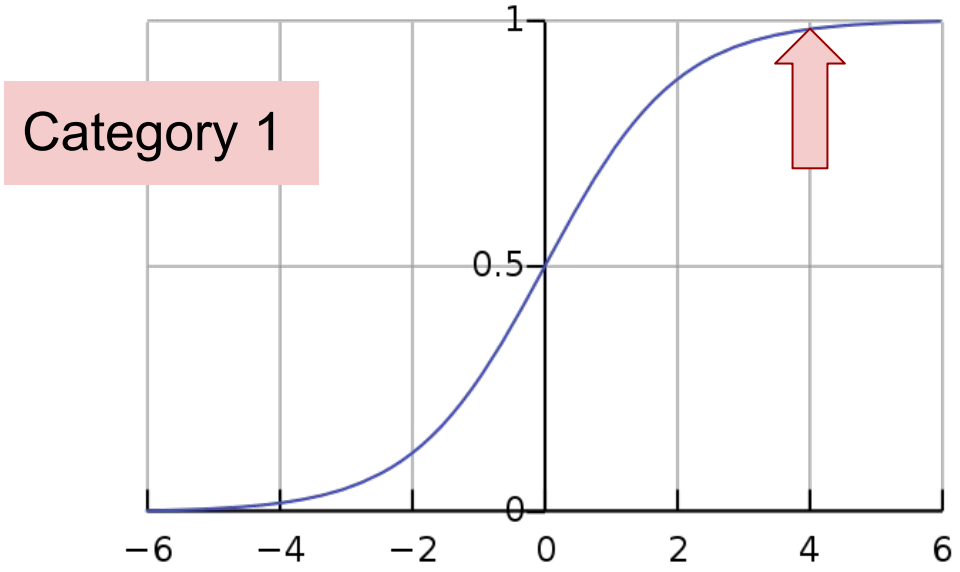
```
sig <- function(x) {
  1/(1 + exp(-x))
}

plot(sigmoid, -6, 6)
```



The Logistic Function

$$\sigma(x) = \frac{1}{1 + e^{-(x)}}$$

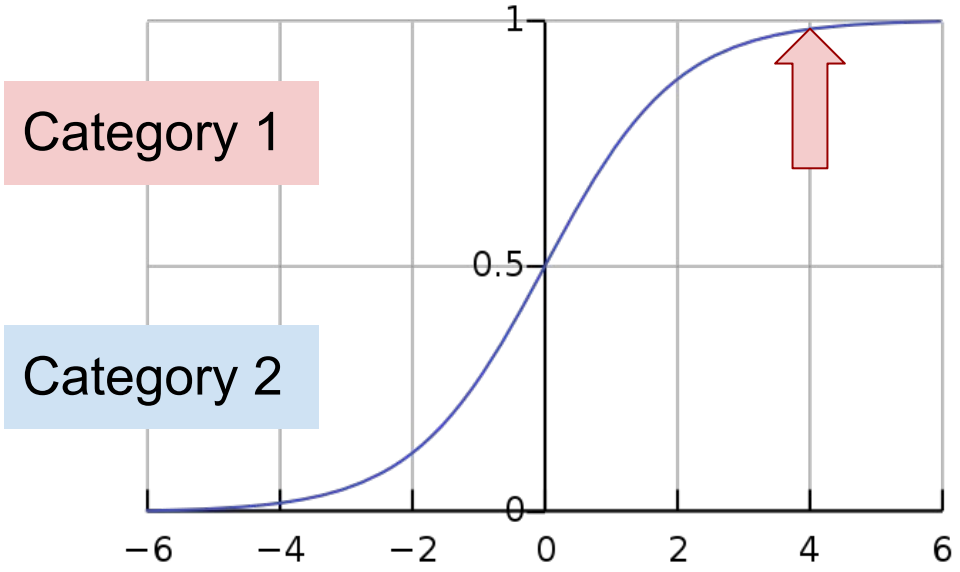
```
sig(n) >= 0.5
```



The Logistic Function

$$\sigma(x) = \frac{1}{1 + e^{-(x)}}$$

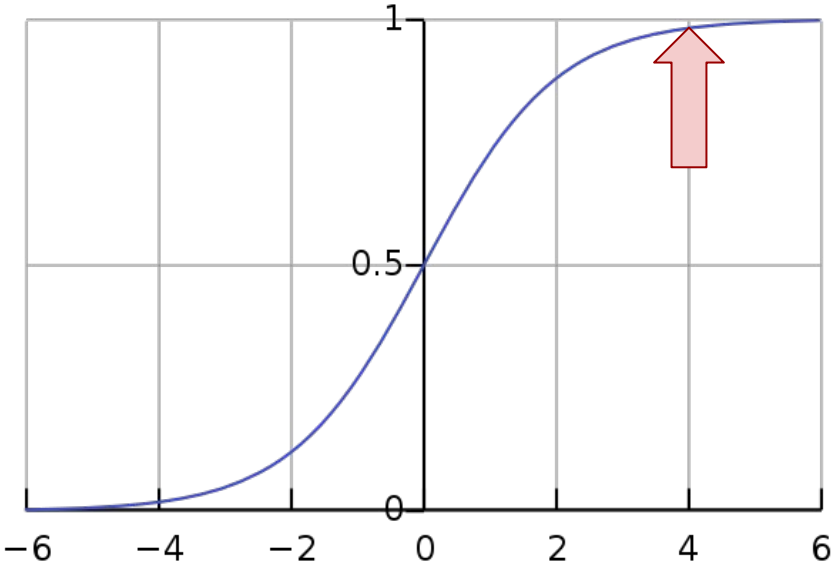
```
sig(n)  < 0.5
```



The Logistic Function

$$\sigma(x) = \frac{1}{1 + e^{-(x)}}$$

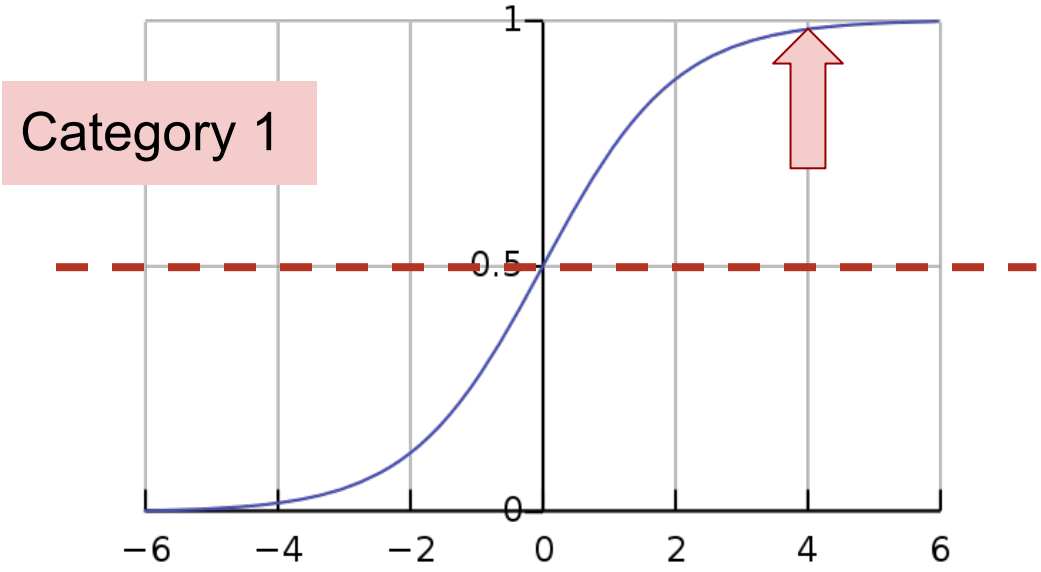
```
sig(4.0)
[1] 0.9820138
```



The Logistic Function

$$\sigma(x) = \frac{1}{1+e^{-(x)}}$$

```
sig(4.0)
[1] 0.9820138
```



Logistic Regression

$$P_{br} = \sigma(C_{Break}Break + C_{BackEdge}BackEdge + C_{EqZero}EqZero + C_{IsZero}IsZero + C_{NoPDom}NoPDom + C_{ToFun}ToFun + C)$$

Logistic Regression

$(C_{Break}, C_{BackEdge}, C_{EqZero}, C_{IsZero}, C_{NoPDom}, C_{ToFun})$

$$P_{br} = \sigma(C_{Break}Break + C_{BackEdge}BackEdge + C_{EqZero}EqZero + C_{IsZero}IsZero + C_{NoPDom}NoPDom + C_{ToFun}ToFun + C)$$

Logistic Regression

$$(C_{Break}, C_{BackEdge}, C_{EqZero}, C_{IsZero}, C_{NoPDom}, C_{ToFun}) \times \begin{matrix} (Break \\ BackEdge \\ EqZero \\ IsZero \\ NoPDom \\ ToFun) \end{matrix}$$

$$P_{br} = \sigma(C_{Break}Break + C_{BackEdge}BackEdge + C_{EqZero}EqZero + C_{IsZero}IsZero + C_{NoPDom}NoPDom + C_{ToFun}ToFun + C)$$

Logistic Regression

$$n = (C_{Break}, C_{BackEdge}, C_{EqZero}, C_{IsZero}, C_{NoPDom}, C_{ToFun}) \times (Break\ BackEdge\ EqZero\ IsZero\ NoPDom\ ToFun) + C$$

$$P_{br} = \sigma(C_{Break}Break + C_{BackEdge}BackEdge + C_{EqZero}EqZero + C_{IsZero}IsZero + C_{NoPDom}NoPDom + C_{ToFun}ToFun + C)$$

Logistic Regression

$$n = (C_{Break}, C_{BackEdge}, C_{EqZero}, C_{IsZero}, C_{NoPDom}, C_{ToFun}) \times \begin{pmatrix} Break \\ BackEdge \\ EqZero \\ IsZero \\ NoPDom \\ ToFun \end{pmatrix} + C$$

$$P_{br} = \sigma(n)$$

$$P_{br} = \sigma(C_{Break}Break + C_{BackEdge}BackEdge + C_{EqZero}EqZero + C_{IsZero}IsZero + C_{NoPDom}NoPDom + C_{ToFun}ToFun + C)$$

Logistic Regression

$$n = (C_{Break}, C_{BackEdge}, C_{EqZero}, C_{IsZero}, C_{NoPDom}, C_{ToFun}) \times \begin{matrix} (Break \\ BackEdge \\ EqZero \\ IsZero \\ NoPDom \\ ToFun) \end{matrix} + C$$

$$P_{br} = \sigma(n)$$

if (P_{br}) >= 0.5 then "Taken" else "Not Taken"

Branch Features

Break

BackEdge

EqZero

IsZero

NoPDom

ToFun

Branch Features

- Binary

Break

BackEdge

EqZero

IsZero

NoPDom

ToFun

Branch Features

- Binary
- Static

Break

BackEdge

EqZero

IsZero

NoPDom

ToFun

Branch Features

- Binary
- Static
- Deterministic

Break

BackEdge

EqZero

IsZero

NoPDom

ToFun

Branch Features

- Binary
- Static
- Deterministic
- Available

Break

BackEdge

EqZero

IsZero

NoPDom

ToFun

Features

- Binary
- Static
- Deterministic
- Available

Fernando Magno Quintão Pereira, Guilherme V. Leobas, Abdoulaye Gamatié:
Static Prediction of Silent Stores.
ACM Trans. Archit. Code Optim. 15(4): 44:1-44:26 (2019)

Break

BackEdge

EqZero

IsZero

NoPDom

ToFun

Break

BackEdge

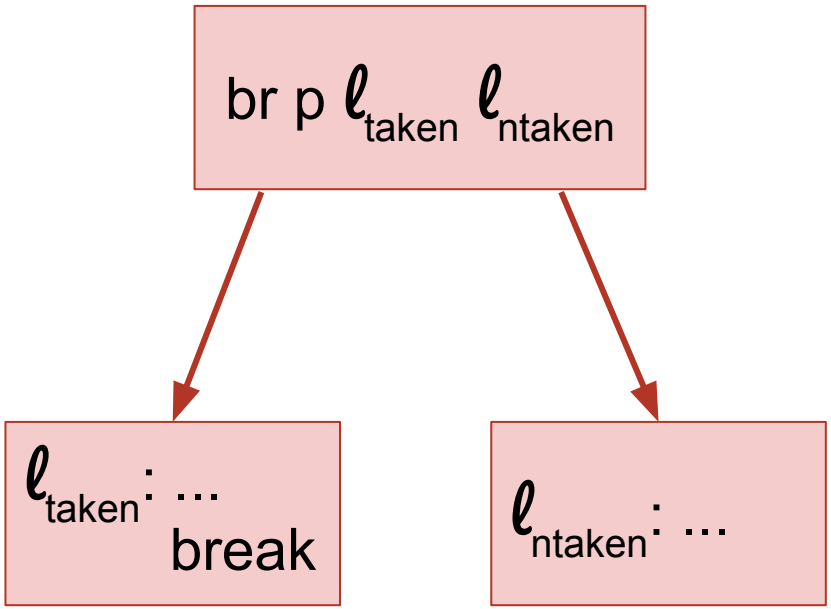
EqZero

IsZero

NoPDom

ToFun

Angélica Aparecida Moreira, Guilherme Ottoni, Fernando Magno Quintão Pereira:
VESPA: static profiling for binary optimization.
Proc. ACM Program. Lang. 5(OOPSLA): 1-28 (2021)



Break

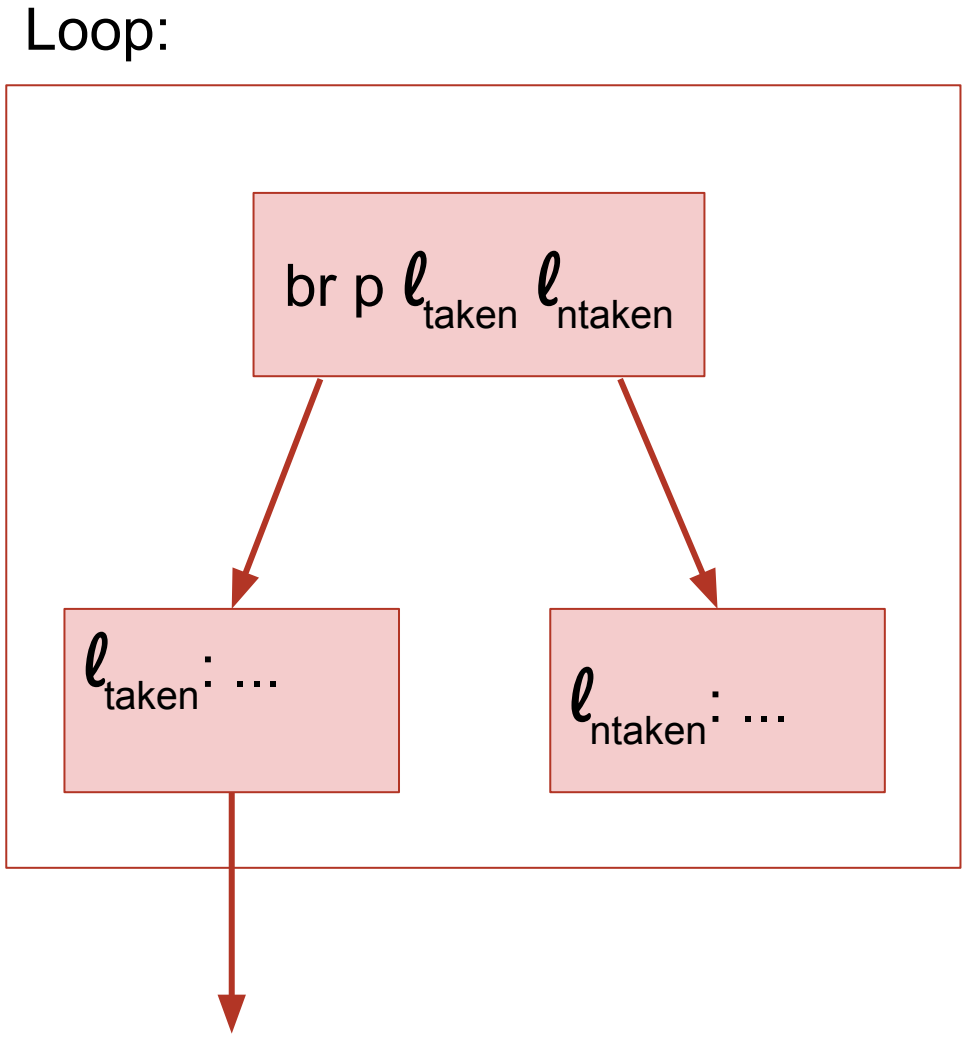
BackEdge

EqZero

IsZero

NoPDom

ToFun)



Break

BackEdge

EqZero

IsZero

NoPDom

ToFun)

(Break

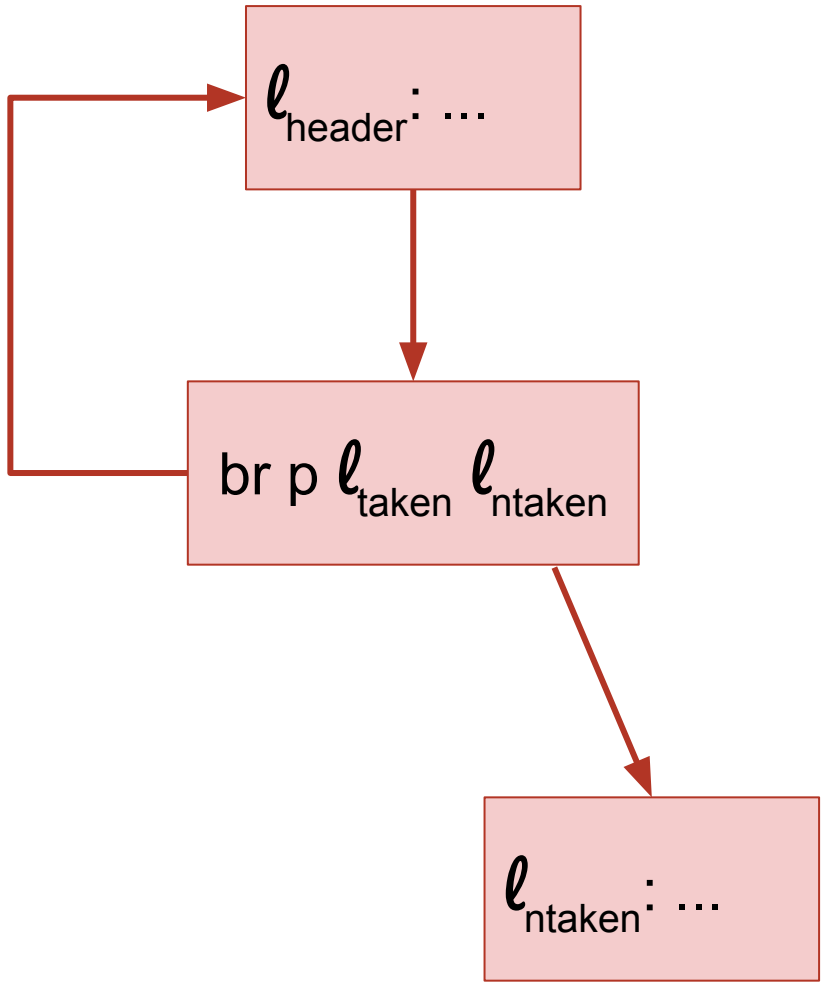
BackEdge

EqZero

IsZero

NoPDom

ToFun)



(Break

BackEdge

EqZero

IsZero

NoPDom

ToFun)

(Break

BackEdge

EqZero

IsZero

NoPDom

ToFun)

p = (**x == 0**)

br p ℓ_{taken} ℓ_{ntaken}

(Break

BackEdge

EqZero

IsZero

NoPDom

ToFun)

p = (**x == 0**)

br p ℓ_{taken} ℓ_{ntaken}

p = (**0 == x**)

br p ℓ_{taken} ℓ_{ntaken}

(Break

BackEdge

EqZero

IsZero

NoPDom

ToFun)

p = (**x == 0**)

br p ℓ_{taken} ℓ_{ntaken}

p = (**0 == x**)

br p ℓ_{taken} ℓ_{ntaken}

p = (**0 == x**)

q = !p

br p ℓ_{taken} ℓ_{ntaken}

(Break

BackEdge

EqZero

IsZero

NoPDom

ToFun)

(Break

BackEdge

EqZero

IsZero

NoPDom

ToFun)

(Break

BackEdge

EqZero

NonConstantOperandCouldBeZero ~~IsZero~~

NoPDom

ToFun)

```
int foo(int x, int* z) {
  int sum = 0;
  for (int i = 0; i < x; i++) {
    if (z[i] == 0) {
      sum += z[0];
    }
  }
  return sum;
}
```

(Break

BackEdge

EqZero

IsZero

NoPDom

ToFun)

What are the branches in this example?

```
int foo(int x, int* z) {
  int sum = 0;
  for (int i = 0; i < x; i++) {
    if (z[i] == 0) {
      sum += z[0];
    }
  }
  return sum;
}
```

(Break

BackEdge

EqZero

IsZero

NoPDom

ToFun)

```
int foo(int x, int* z) {
  int sum = 0;
  for (int i = 0; i < x; i++) {
    if (z[i] == 0) {
      sum += z[0];
    }
  }
  return sum;
}
```

(Break

BackEdge

EqZero

IsZero

NoPDom

ToFun)

```
int foo(int x, int* z) {
  int sum = 0;
  for (int i = 0; i < x; i++) {
    if (z[i] == 0) {
      sum += z[0];
    }
  }
  return sum;
}
```

(Break

BackEdge

EqZero

IsZero

NoPDom

ToFun)

```
int foo(int x, int* z) {
  int sum = 0;
  for (int i = 0; i < x; i++) {
    if (z[i] == 0) {
      sum += z[0];
    }
  }
  return sum;
}
```

- Static Analysis

(Break

BackEdge

EqZero

IsZero

NoPDom

ToFun)

```
int foo(int x, int* z) {
  int sum = 0;
  for (int i = 0; i < x; i++) {
    if (z[i] == 0) {
      sum += z[0];
    }
  }
  return sum;
}
```

(Break

BackEdge

EqZero

IsZero

NoPDom

ToFun)

- Static Analysis

Can you think about a static analysis to compute this feature?

```
int foo(int x, int* z) {
  int sum = 0;
  for (int i = 0; i < x; i++) {
    if (z[i] == 0) {
      sum += z[0];
    }
  }
  return sum;
}
```

(Break

BackEdge

EqZero

IsZero

NoPDom

ToFun)

- Static Analysis: backward slice

Break

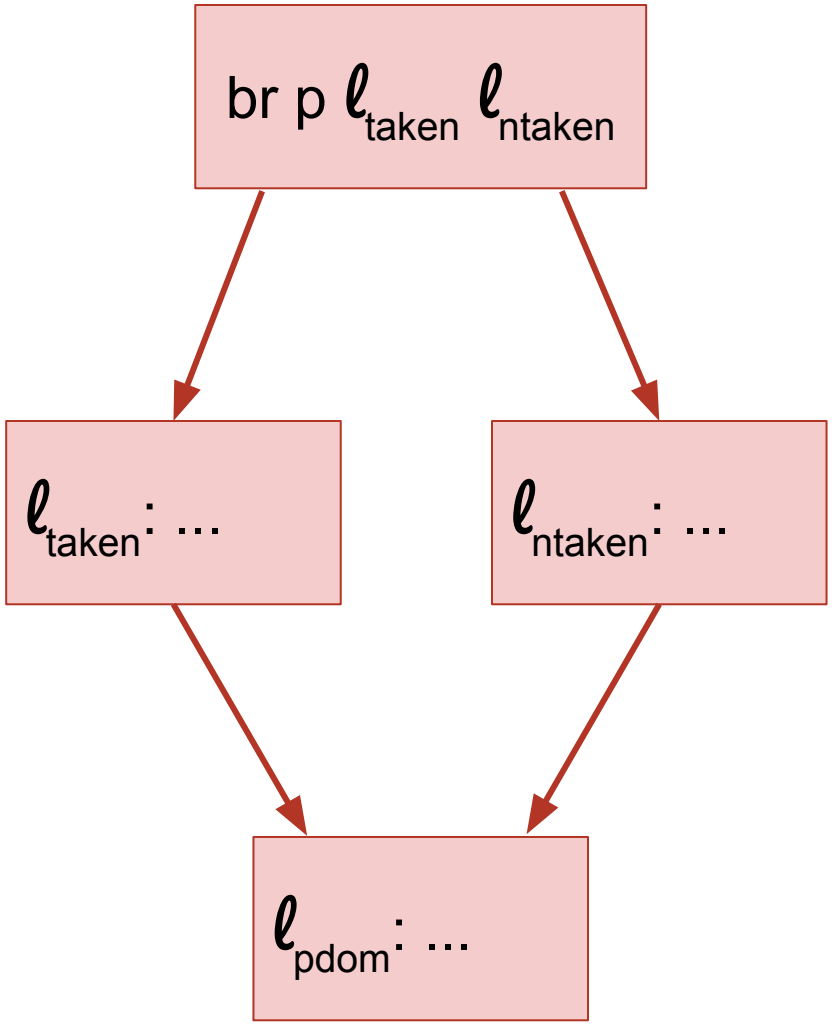
BackEdge

EqZero

IsZero

NoPDom

ToFun)



Break

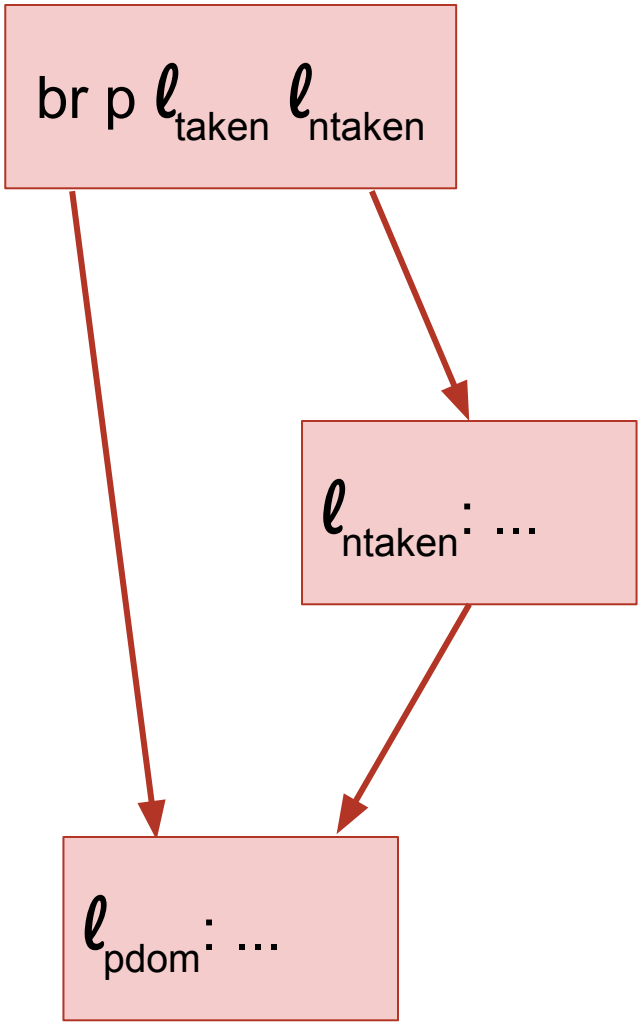
BackEdge

EqZero

IsZero

1 == NoPDom

ToFun)



Break

BackEdge

EqZero

IsZero

0 == NoPDom

ToFun)

(Break

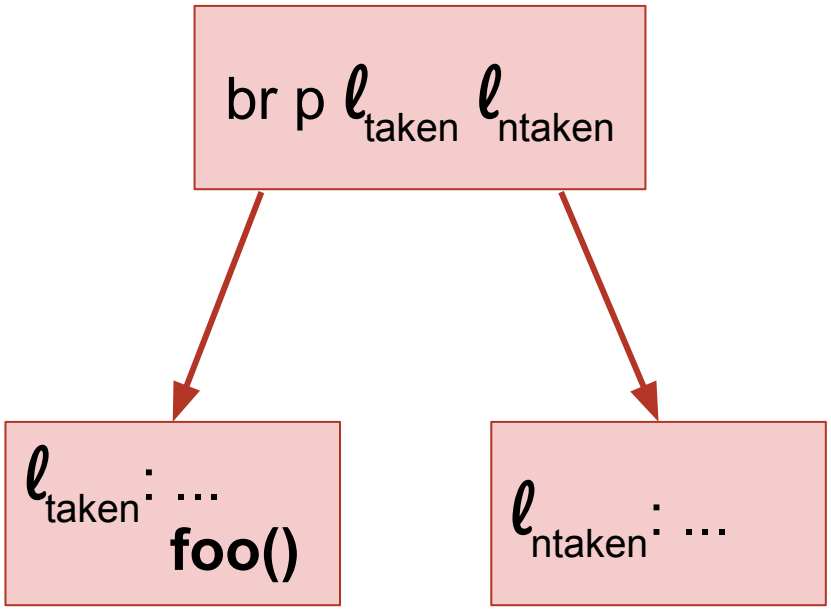
BackEdge

EqZero

IsZero

NoPDom

ToFun



(Break

BackEdge

EqZero

IsZero

NoPDom

ToFun

Break

BackEdge

EqZero

IsZero

NoPDom

ToFun

Angélica Aparecida Moreira, Guilherme Ottoni, Fernando Magno Quintão Pereira:
VESPA: static profiling for binary optimization.
Proc. ACM Program. Lang. 5(OOPSLA): 1-28 (2021)

Example: Logistic Classification in R

Example: Logistic Classification in R

https://www.dcc.ufmg.br/~fernando/classes/dcc888/ementa/data_ml/branch_features.csv

Example: Logistic Classification in R

https://www.dcc.ufmg.br/~fernando/classes/dcc888/ementa/data_ml/branch_features.csv

Branch	Break	BackEdge	EqZero	IsZero	NoPDom	ToFun	Freq	Taken
B1	0	0	0	0	0	0	1	0
B2	0	0	0	0	0	1	63	0
B3	0	0	0	0	1	0	51	0
B4	0	0	0	0	1	1	113	0
B5	0	0	0	1	0	0	46	0
B6	0	0	0	1	0	1	108	0
B7	0	0	0	1	1	0	96	0
B8	0	0	0	1	1	1	158	1
B9	0	0	1	0	0	0	19	0
B10	0	0	1	0	0	1	81	0
B11	0	0	1	0	1	0	69	0

Example: Logistic Classification in R

https://www.dcc.ufmg.br/~fernando/classes/dcc888/ementa/data_ml/branch_features.csv

Branch	Break	BackEdge	EqZero	IsZero	NoPDom	ToFun	Freq	Taken
B1	0	0	0	0	0	0	1	0
B2	0	0	0	0	0	1	63	0
B3	0	0	0	0	1	0	51	0
B4	0	0	0	0	1	1	113	0
B5	0	0	0	1	0	0	46	0
B6	0	0	0	1	0	1	108	0
B7	0	0	0	1	1	0	96	0
B8	0	0	0	1	1	1	158	1
B9	0	0	1	0	0	0	19	0
B10	0	0	1	0	0	1	81	0
B11	0	0	1	0	1	0	69	0

Example: Logistic Classification in R

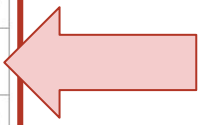
https://www.dcc.ufmg.br/~fernando/classes/dcc888/ementa/data_ml/branch_features.csv

Branch	Break	BackEdge	EqZero	IsZero	NoPDom	ToFun	Freq	Taken
B1	0	0	0	0	0	0	1	0
B2	0	0	0	0	0	1	63	0
B3	0	0	0	0	1	0	51	0
B4	0	0	0	0	1	1	113	0
B5	0	0	0	1	0	0	46	0
B6	0	0	0	1	0	1	108	0
B7	0	0	0	1	1	0	96	0
B8	0	0	0	1	1	1	158	1
B9	0	0	1	0	0	0	19	0
B10	0	0	1	0	0	1	81	0
B11	0	0	1	0	1	0	69	0

Example: Logistic Classification in R

https://www.dcc.ufmg.br/~fernando/classes/dcc888/ementa/data_ml/branch_features.csv

Branch	Break	BackEdge	EqZero	IsZero	NoPDom	ToFun	Freq	Taken
B1	0	0	0	0	0	0	1	0
B2	0	0	0	0	0	1	63	0
B3	0	0	0	0	1	0	51	0
B4	0	0	0	0	1	1	113	0
B5	0	0	0	1	0	0	46	0
B6	0	0	0	1	0	1	108	0
B7	0	0	0	1	1	0	96	0
B8	0	0	0	1	1	1	158	1
B9	0	0	1	0	0	0	19	0
B10	0	0	1	0	0	1	81	0
B11	0	0	1	0	1	0	69	0



Example: Logistic Classification in R

```
data <- read.csv("~/branch_features.csv")

shuffled_data <- data[sample(1:nrow(data)), ]

num_samples <- nrow(shuffled_data)

train_size <- ceiling(num_samples*0.8)

train <- shuffled_data[1:train_size,]

test <- shuffled_data[(train_size+1):num_samples,]

model <- glm(Taken ~ Break + BackEdge + EqZero + IsZero +
NoPDom + ToFun, family=binomial(link='logit'), data=train)

results <- predict(model, test, type="response")

predictions <- ifelse(results > 0.5, 1, 0)

misses <- mean(predictions != test$Taken)

accuracy <- 1 - misses
```

Example: Logistic Classification in R

```
data <- read.csv("~/branch_features.csv")
shuffled_data <- data[sample(1:nrow(data)), ]
num_samples <- nrow(shuffled_data)
train_size <- ceiling(num_samples*0.8)
train <- shuffled_data[1:train_size,]
test <- shuffled_data[(train_size+1):num_samples,]
model <- glm(Taken ~ Break + BackEdge + EqZero + IsZero +
NoPDom + ToFun, family=binomial(link='logit'), data=train)
results <- predict(model, test, type="response")
predictions <- ifelse(results > 0.5, 1, 0)
misses <- mean(predictions != test$Taken)
accuracy <- 1 - misses
```

Example: Logistic Classification in R

```
data <- read.csv("~/branch_features.csv")
shuffled_data <- data[sample(1:nrow(data)), ]
num_samples <- nrow(shuffled_data)
train_size <- ceiling(num_samples*0.8)
train <- shuffled_data[1:train_size,]
test <- shuffled_data[(train_size+1):num_samples,]
model <- glm(Taken ~ Break + BackEdge + EqZero + IsZero +
NoPDom + ToFun, family=binomial(link='logit'), data=train)
results <- predict(model, test, type="response")
predictions <- ifelse(results > 0.5, 1, 0)
misses <- mean(predictions != test$Taken)
accuracy <- 1 - misses
```

Example: Logistic Classification in R

```
data <- read.csv("~/branch_features.csv")
shuffled_data <- data[sample(1:nrow(data)), ]
num_samples <- nrow(shuffled_data)
```

Branch	Break	BackEdge	EqZero	IsZero	NoPDom	ToFun	Freq	Taken
B1	0	0	0	0	0	0	1	0
B2	0	0	0	0	0	1	63	0
B3	0	0	0	0	1	0	51	0
B4	0	0	0	0	1	1	113	0
B5	0	0	0	1	0	0	46	0
B6	0	0	0	1	0	1	108	0
B7	0	0	0	1	1	0	96	0
B8	0	0	0	1	1	1	158	1
B9	0	0	1	0	0	0	19	0
B10	0	0	1	0	0	1	81	0
B11	0	0	1	0	1	0	69	0

Example: Logistic Classification in R

```
data <- read.csv("~/branch_features.csv")
shuffled_data <- data[sample(1:nrow(data)), ]
num_samples <- nrow(shuffled_data)
```

Branch	Break	BackEdge	EqZero	IsZero	NoPDom	ToFun	Freq	Taken
B1	0	0	0	0	0	0	1	0
B2	0	0	0	0	0	1	63	0
B3	0	0	0	0	1	0	51	0
B4	0	0	0	0	1	1	113	0
B5	0	0	0	1	0	0	46	0
B6	0	0	0	1	0	1	108	0
B7	0	0	0	1	1	0	96	0
B8	0	0	0	1	1	1	158	1
B9	0	0	1	0	0	0	19	0
B10	0	0	1	0	0	1	81	0
B11	0	0	1	0	1	0	69	0

Example: Logistic Classification in R

```
data <- read.csv("~/branch_features.csv")

shuffled_data <- data[sample(1:nrow(data)), ]

num_samples <- nrow(shuffled_data)

train_size <- ceiling(num_samples*0.8)

train <- shuffled_data[1:train_size,]

test <- shuffled_data[(train_size+1):num_samples,]

model <- glm(Taken ~ Break + BackEdge + EqZero + IsZero +
NoPDom + ToFun, family=binomial(link='logit'), data=train)

results <- predict(model, test, type="response")

predictions <- ifelse(results > 0.5, 1, 0)

misses <- mean(predictions != test$Taken)

accuracy <- 1 - misses
```

Example: Logistic Classification in R

```
data <- read.csv("~/branch_features.csv")
shuffled_data <- data[sample(1:nrow(data)), ]
num_samples <- nrow(shuffled_data)
train_size <- ceiling(num_samples*0.8)
train <- shuffled_data[1:train_size,]
test <- shuffled_data[(train_size+1):num_samples,]
model <- glm(Taken ~ Break + BackEdge + EqZero + IsZero +
NoPDom + ToFun, family=binomial(link='logit'), data=train)
results <- predict(model, test, type="response")
predictions <- ifelse(results > 0.5, 1, 0)
misses <- mean(predictions != test$Taken)
accuracy <- 1 - misses
```

Example: Logistic Classification in R

```
data <- read.csv("~/branch_features.csv")
shuffled_data <- data[sample(1:nrow(data)), ]
num_samples <- nrow(shuffled_data)
train_size <- ceiling(num_samples*0.8)
train <- shuffled_data[1:train_size,]
test <- shuffled_data[(train_size+1):num_samples,]
model <- glm(Taken ~ Break + BackEdge + EqZero + IsZero +
NoPDom + ToFun, family=binomial(link='logit'), data=train)
results <- predict(model, test, type="response")
predictions <- ifelse(results > 0.5, 1, 0)
misses <- mean(predictions != test$Taken)
accuracy <- 1 - misses
```

Example: Logistic Classification in R

```
data <- read.csv("~/branch_features.csv")
shuffled_data <- data[sample(1:nrow(data)), ]
num_samples <- nrow(shuffled_data)
train_size <- ceiling(num_samples*0.8)
train <- shuffled_data[1:train_size,]
test <- shuffled_data[(train_size+1):num_samples,]
model <- glm(Taken ~ Break + BackEdge + EqZero + IsZero +
NoPDom + ToFun, family=binomial(link='logit'), data=train)
results <- predict(model, test, type="response")
predictions <- ifelse(results > 0.5, 1, 0)
misses <- mean(predictions != test$Taken)
accuracy <- 1 - misses
```

Example: Logistic Classification in R

```
data <- read.csv("~/branch_features.csv")
shuffled_data <- data[sample(1:nrow(data)), ]
num_samples <- nrow(shuffled_data)
train_size <- ceiling(num_samples*0.8)
train <- shuffled_data[1:train_size,]
test <- shuffled_data[(train_size+1):num_samples,]
model <- glm(Taken ~ Break + BackEdge + EqZero + IsZero +
NoPDom + ToFun, family=binomial(link='logit'), data=train)
results <- predict(model, test, type="response")
predictions <- ifelse(results > 0.5, 1, 0)
misses <- mean(predictions != test$Taken)
accuracy <- 1 - misses
```

Example: Logistic Classification in R

```
data <- read.csv("~/branch_features.csv")
shuffled_data <- data[sample(1:nrow(data)), ]
num_samples <- nrow(shuffled_data)
train_size <- ceiling(num_samples*0.8)
train <- shuffled_data[1:train_size,]
test <- shuffled_data[(train_size+1):num_samples,]
model <- glm(Taken ~ Break + BackEdge + EqZero + IsZero +
NoPDom + ToFun, family=binomial(link='logit'), data=train)
results <- predict(model, test, type="response")
predictions <- ifelse(results > 0.5, 1, 0)
misses <- mean(predictions != test$Taken)
accuracy <- 1 - misses
```

Example: Logistic Classification in R

```
data <- read.csv("~/branch_features.csv")
shuffled_data <- data[sample(1:nrow(data)), ]
num_samples <- nrow(shuffled_data)
train_size <- ceiling(num_samples*0.8)
train <- shuffled_data[1:train_size,]
test <- shuffled_data[(train_size+1):num_samples,]
model <- glm(Taken ~ Break + BackEdge + EqZero + IsZero +
NoPDom + ToFun, family=binomial(link='logit'), data=train)
results <- predict(model, test, type="response")
predictions <- ifelse(results > 0.5, 1, 0)
misses <- mean(predictions != test$Taken)
accuracy <- 1 - misses
```

Example: Logistic Classification in R

```
data <- read.csv("~/branch_features.csv")
shuffled_data <- data[sample(1:nrow(data)), ]
num_samples <- nrow(shuffled_data)
train_size <- ceiling(num_samples*0.8)
train <- shuffled_data[1:train_size,]
test <- shuffled_data[(train_size+1):num_samples,]
model <- glm(Taken ~ Break + BackEdge + EqZero + IsZero +
NoPDom + ToFun, family=binomial(link='logit'), data=train)
results <- predict(model, test, type="response")
predictions <- ifelse(results > 0.5, 1, 0)
misses <- mean(predictions != test$Taken)
accuracy <- 1 - misses
```

Example: Logistic Classification in R

```
data <- read.csv("~/branch_features.csv")
shuffled_data <- data[sample(1:nrow(data)), ]
num_samples <- nrow(shuffled_data)
train_size <- ceiling(num_samples*0.8)
train <- shuffled_data[1:train_size,]
test <- shuffled_data[(train_size+1):num_samples,]
model <- glm(Taken ~ Break + BackEdge + EqZero + IsZero +
NoPDom + ToFun, family=binomial(link='logit'), data=train)
results <- predict(model, test, type="response")
predictions <- ifelse(results > 0.5, 1, 0)
misses <- mean(predictions != test$Taken)
accuracy <- 1 - misses
```

Example: Logistic Classification in R

```
data <- read.csv("~/branch_features.csv")

shuffled_data <- data[sample(1:nrow(data)), ]

num_samples <- nrow(shuffled_data)

train_size <- ceiling(num_samples*0.8)

train <- shuffled_data[1:train_size,]

test <- shuffled_data[(train_size+1):num_samples,]

model <- glm(Taken ~ Break + BackEdge + EqZero + IsZero +
NoPDom + ToFun, family=binomial(link='logit'), data=train)

results <- predict(model, test, type="response")

predictions <- ifelse(results > 0.5, 1, 0)

misses <- mean(predictions != test$Taken)

accuracy <- 1 - misses
```

Example: Logistic Classification in R

```
data <- read.csv("~/branch_features.csv")
shuffled_data <- data[sample(1:nrow(data)), ]
num_samples <- nrow(shuffled_data)
train_size <- ceiling(num_samples*0.8)
train <- shuffled_data[1:train_size,]
test <- shuffled_data[(train_size+1):num_samples,]
model <- glm(Taken ~ Break + BackEdge + EqZero + IsZero +
NoPDom + ToFun, family=binomial(link='logit'), data=train)
results <- predict(model, test, type="response")
predictions <- ifelse(results > 0.5, 1, 0)
misses <- mean(predictions != test$Taken)
accuracy <- 1 - misses
```

Example: Logistic Classification in R

```
data <- read.csv("~/branch_features.csv")

shuffled_data <- data[sample(1:nrow(data)), ]

num_samples <- nrow(shuffled_data)

train_size <- ceiling(num_samples*0.8)

train <- shuffled_data[1:train_size,]

test <- shuffled_data[(train_size+1):num_samples,]

model <- glm(Taken ~ Break + BackEdge + EqZero + IsZero +
NoPDom + ToFun, family=binomial(link='logit'), data=train)

results <- predict(model, test, type="response")

predictions <- ifelse(results > 0.5, 1, 0)

misses <- mean(predictions != test$Taken)

accuracy <- 1 - misses
```

0.9166

Example: Logistic Classification in R

```
data <- read.csv("~/branch_features.csv")

shuffled_data <- data[sample(1:nrow(data)), ]

num_samples <- nrow(shuffled_data)

train_size <- ceiling(num_samples*0.6)

train <- shuffled_data[1:train_size,]

test <- shuffled_data[(train_size+1):num_samples,]

model <- glm(Taken ~ Break + BackEdge + EqZero + IsZero +
NoPDom + ToFun, family=binomial(link='logit'), data=train)

results <- predict(model, test, type="response")

predictions <- ifelse(results > 0.5, 1, 0)

misses <- mean(predictions != test$Taken)

accuracy <- 1 - misses
```

0.8400

Looking into the Model

```
summary(model)
```

Looking into the Model

```
summary(model)
```

```
Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  -242.03   141368.40  -0.002   0.999
Break        -88.19     57995.29  -0.002   0.999
BackEdge     177.13    105428.63   0.002   0.999
EqZero       131.67     81713.77   0.002   0.999
IsZero       133.52     84777.53   0.002   0.999
NoPDom       87.37      57022.02   0.002   0.999
ToFun        86.82      61439.50   0.001   0.999
```

Looking into the Model

```
summary(model)
```

```
Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  -242.03   141368.40  -0.002   0.999
Break         -88.19    57995.29  -0.002   0.999
BackEdge      177.13   105428.63   0.002   0.999
EqZero        131.67    81713.77   0.002   0.999
IsZero        133.52    84777.53   0.002   0.999
NoPDom         87.37    57022.02   0.002   0.999
ToFun          86.82    61439.50   0.001   0.999
```

Looking into the Model

```
summary(model)
```

```
Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  -242.03   141368.40  -0.002   0.999
Break         -88.19    57995.29  -0.002   0.999
BackEdge      177.13   105428.63   0.002   0.999
EqZero        131.67    81713.77   0.002   0.999
IsZero        133.52    84777.53   0.002   0.999
NoPDom         87.37    57022.02   0.002   0.999
ToFun          86.82    61439.50   0.001   0.999
```

$$P_{br} = \sigma(C_{Break}\text{Break} + C_{BackEdge}\text{BackEdge} + C_{EqZero}\text{EqZero} + C_{IsZero}\text{IsZero} + C_{NoPDom}\text{NoPDom} + C_{ToFun}\text{ToFun} + C)$$

Looking into the Model

```
summary(model)
```

```
Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  -242.03   141368.40  -0.002   0.999
Break         -88.19    57995.29  -0.002   0.999
BackEdge      177.13   105428.63   0.002   0.999
EqZero        131.67    81713.77   0.002   0.999
IsZero        133.52    84777.53   0.002   0.999
NoPDom         87.37    57022.02   0.002   0.999
ToFun          86.82    61439.50   0.001   0.999
```

Can you come up with an interpretation for this negative coefficient?

$$P_{br} = \sigma(C_{Break}\text{Break} + C_{BackEdge}\text{BackEdge} + C_{EqZero}\text{EqZero} + C_{IsZero}\text{IsZero} + C_{NoPDom}\text{NoPDom} + C_{ToFun}\text{ToFun} + C)$$

Looking into the Model

```
summary(model)
```

```
Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  -242.03   141368.40  -0.002   0.999
Break        -88.19     57995.29  -0.002   0.999
BackEdge     177.13    105428.63   0.002   0.999
EqZero       131.67     81713.77   0.002   0.999
IsZero       133.52     84777.53   0.002   0.999
NoPDom       87.37      57022.02   0.002   0.999
ToFun        86.82      61439.50   0.001   0.999
```

```
int comp(int* s, int* pw, int N) {
  for (int i = 0; i < N; i++) {
    if (s[i] != pw[i]) {
      break;
    }
  }
}
```

Linearity

$$P_{br} = \sigma(C_{Break}Break + C_{BackEdge}BackEdge + C_{EqZero}EqZero + C_{IsZero}IsZero + C_{NoPDom}NoPDom + C_{ToFun}ToFun + C)$$

Linearity

IsZero

EqZero

$$P_{br} = \sigma(C_{Break}Break + C_{BackEdge}BackEdge + C_{EqZero}EqZero + C_{IsZero}IsZero + C_{NoPDom}NoPDom + C_{ToFun}ToFun + C)$$

Linearity

IsZero = 0

EqZero = 1

p = (**x** == 0)

br p ℓ_{taken} ℓ_{ntaken}

$$P_{br} = \sigma(C_{Break}Break + C_{BackEdge}BackEdge + C_{EqZero}EqZero + C_{IsZero}IsZero + C_{NoPDom}NoPDom + C_{ToFun}ToFun + C)$$

Linearity

IsZero = 0

EqZero = 1

p = (**x** == 0)

br p ℓ_{taken} ℓ_{ntaken}

$$P_{br} = \sigma(C_{Break}Break + C_{BackEdge}BackEdge + C_{EqZero}EqZero + C_{IsZero}IsZero + C_{NoPDom}NoPDom + C_{ToFun}ToFun + C_{Comb}IsZero*EqZero + C)$$

Linearity

IsZero = 0

EqZero = 1

p = (**x** == 0)

br p ℓ_{taken} ℓ_{ntaken}

```
model <- glm(Taken ~ Break + BackEdge + EqZero + IsZero +
             NoPDom + ToFun + EqZero * IsZero,
             family=binomial(link='logit'), data=train)
```

$$P_{br} = \sigma(C_{Break}Break + C_{BackEdge}BackEdge + C_{EqZero}EqZero + C_{IsZero}IsZero + C_{NoPDom}NoPDom + C_{ToFun}ToFun + C_{Comb}IsZero*EqZero + C)$$

Other Examples

Other Examples

Angélica Aparecida Moreira, Guilherme Ottoni, Fernando Magno Quintão Pereira:
VESPA: static profiling for binary optimization.
Proc. ACM Program. Lang. 5(OOPSLA): 1-28 (2021)

Other Examples

Brad Calder, Dirk Grunwald, M. Jones, Donald Lindsay, James Martin, Michael Mozer, B. Zorn:
Evidence-Based Static Branch Prediction Using Machine Learning.
ACM Trans. Program. Lang. Syst. 19(1): 188-222 (1997)

Angélica Aparecida Moreira, Guilherme Ottoni, Fernando Magno Quintão Pereira:
VESPA: static profiling for binary optimization.
Proc. ACM Program. Lang. 5(OOPSLA): 1-28 (2021)

Other Examples

John Cavazos, Michael F. P. O'Boyle:
Method-specific dynamic compilation using logistic regression.
OOPSLA 2006: 229-240

Brad Calder, Dirk Grunwald, M. Jones, Donald Lindsay, James Martin, Michael Mozer, B. Zorn:
Evidence-Based Static Branch Prediction Using Machine Learning.
ACM Trans. Program. Lang. Syst. 19(1): 188-222 (1997)

Angélica Aparecida Moreira, Guilherme Ottoni, Fernando Magno Quintão Pereira:
VESPA: static profiling for binary optimization.
Proc. ACM Program. Lang. 5(OOPSLA): 1-28 (2021)