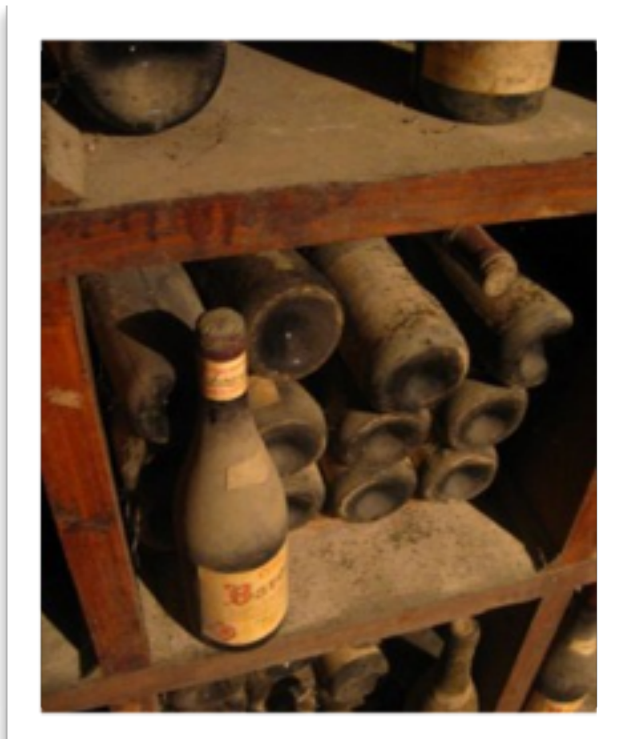
Where Did This Code Come From?

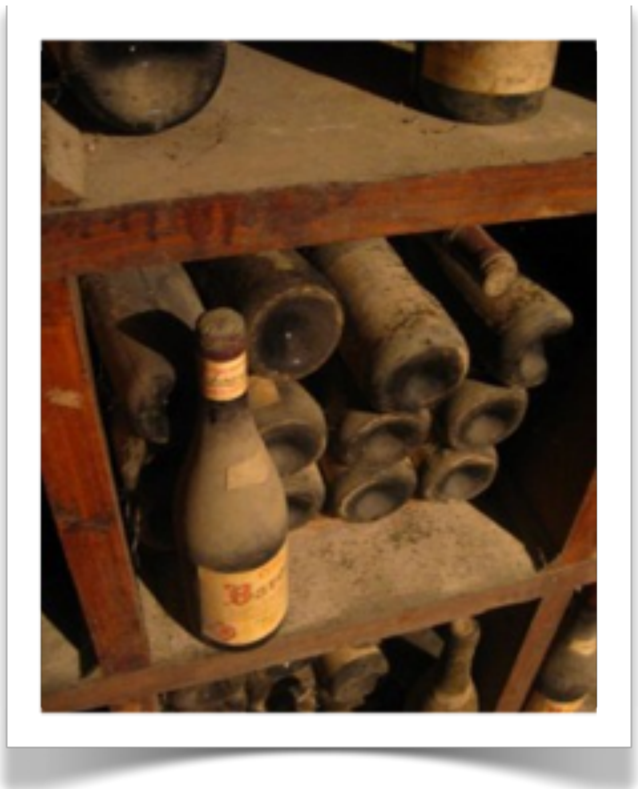
Discovering the provenance of program binaries

Nathan Rosenblum
Computer Sciences Department
University of Wisconsin

nater@cs.wisc.edu
http://pages.cs.wisc.edu/~nater/

Joint work with
Barton Miller and Xiaojin Zhu

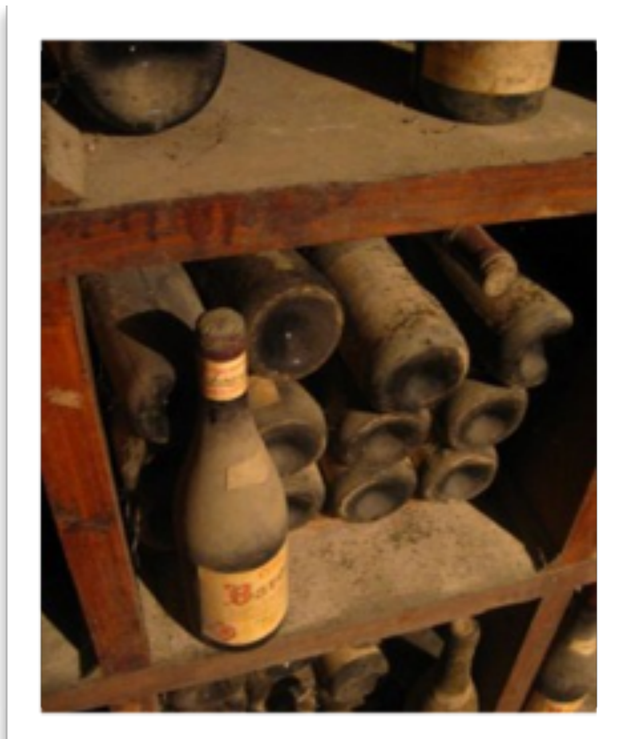




terroir
vineyard
aging
storage

discovery
restoration
posession

focus of this talk



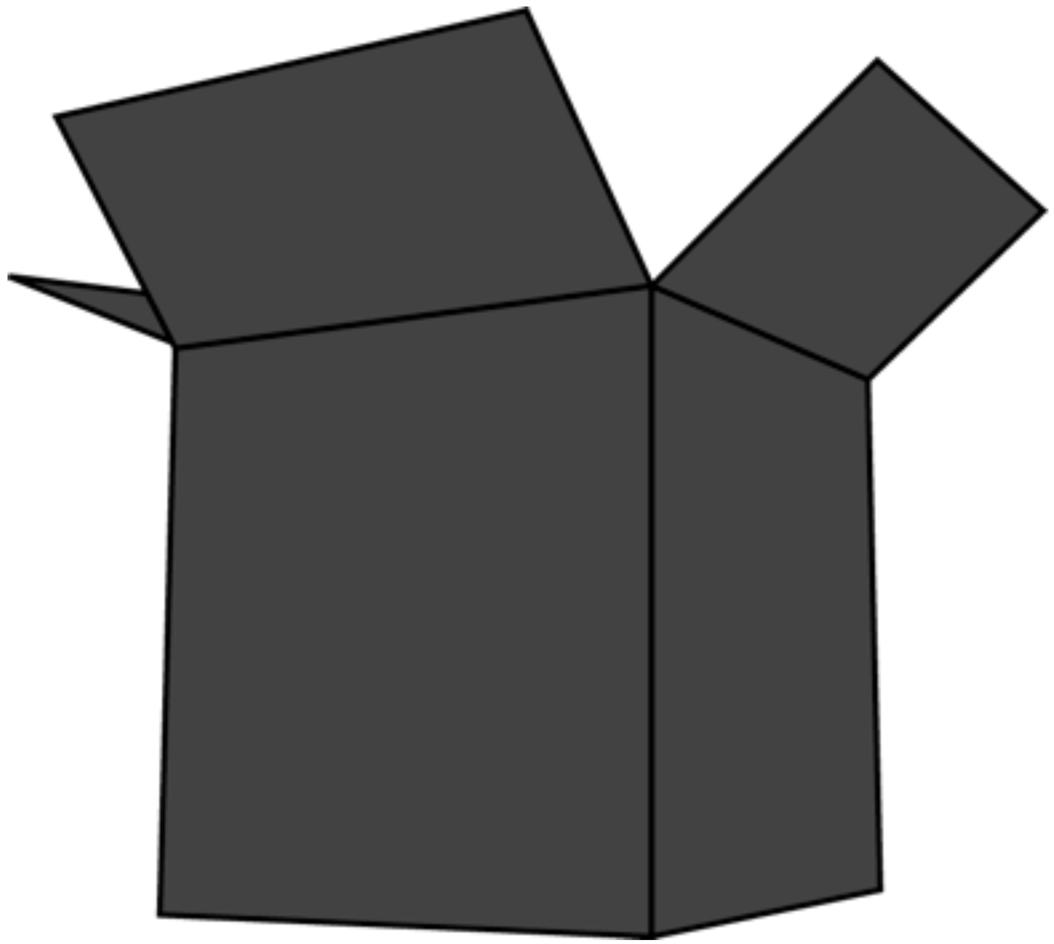
terroir
vineyard
aging
storage

discovery
restoration
posession

Para
dyn™

2

Program provenance



Program provenance

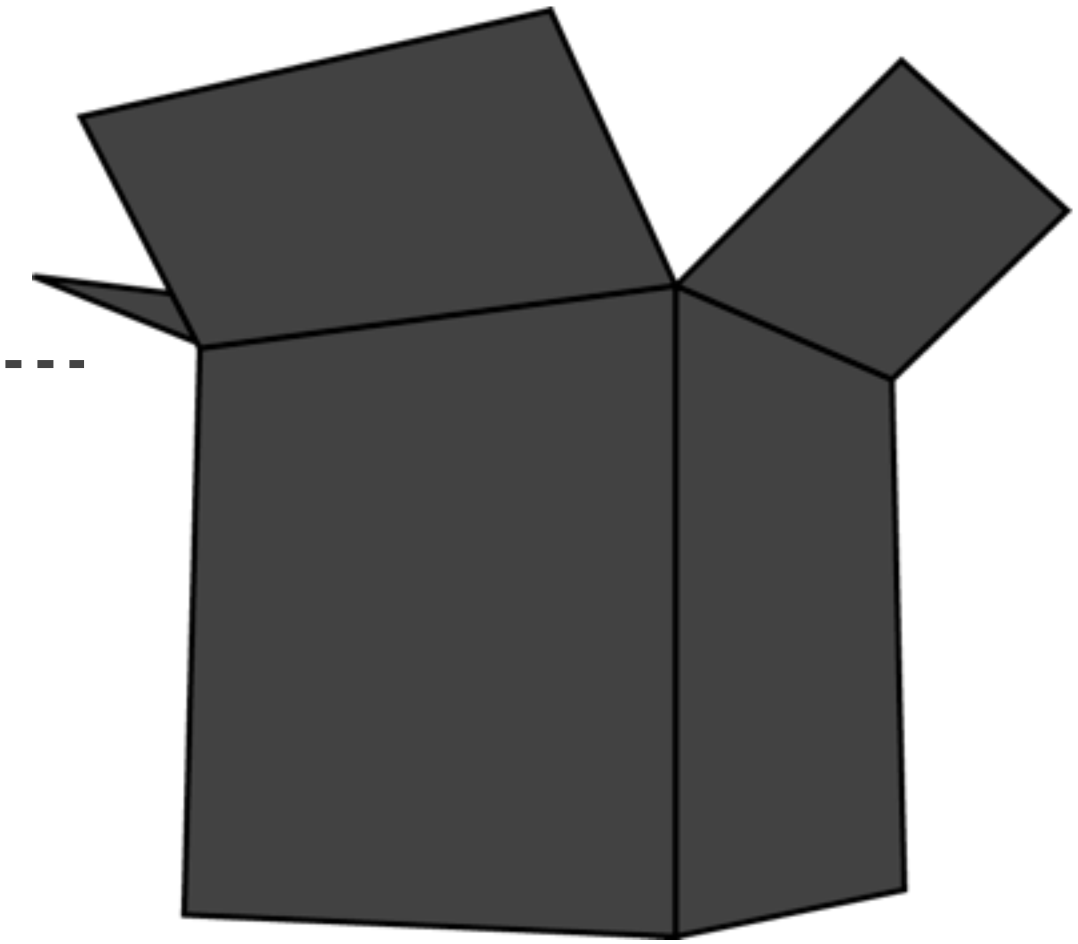
Compiler

family

version

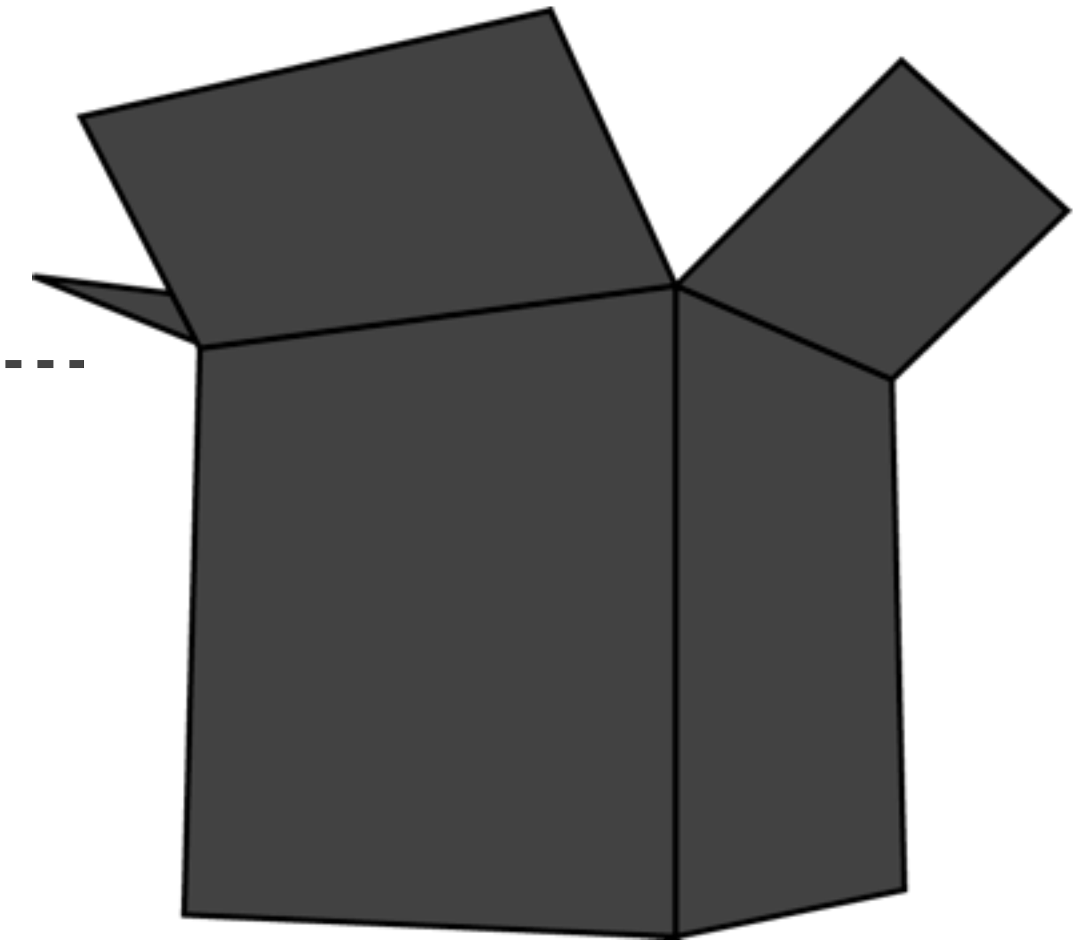
optimization level

source language



Paradyn™

Program provenance



Compiler

family

version

optimization level

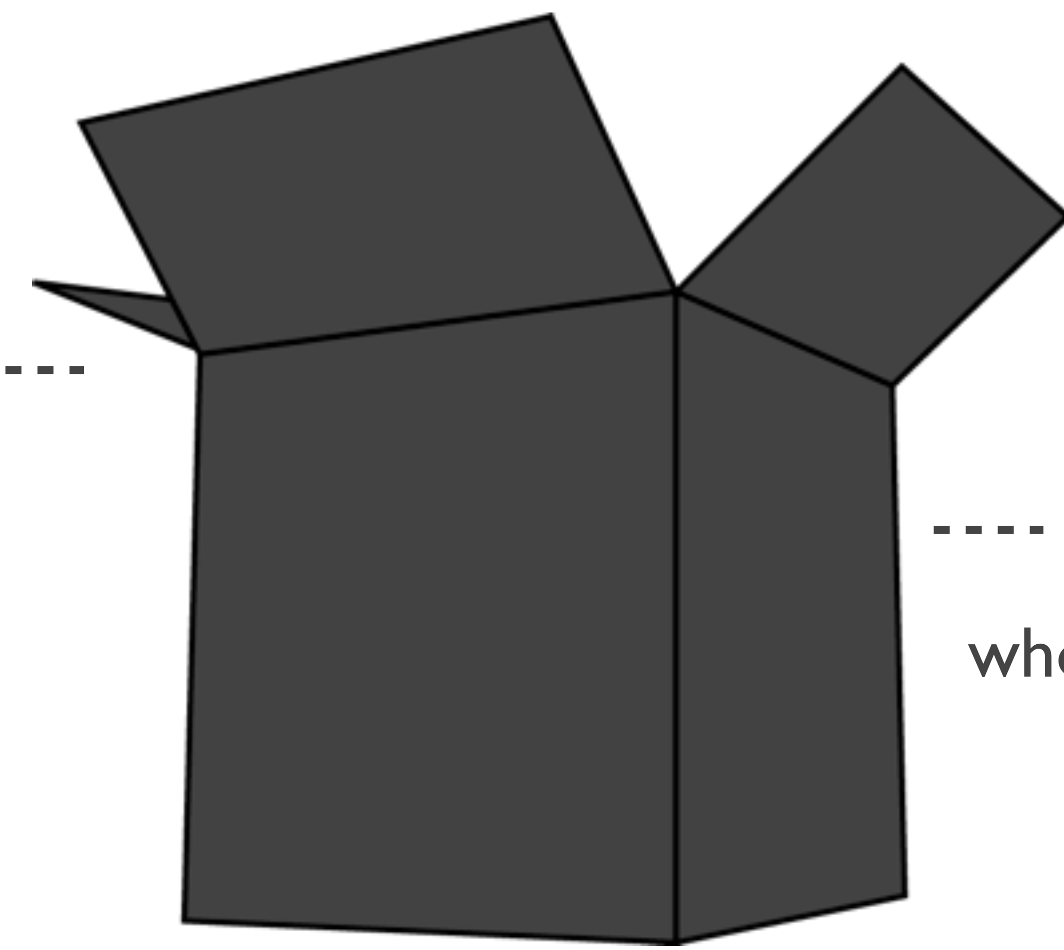
source language

System

glibc static code

library imports

Program provenance



Compiler

family

version

optimization level

source language

Link & post-link

whole-program optimization

rewriting tools

obfuscation tools

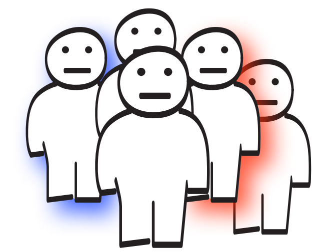
System

glibc static code

library imports

Program provenance

Authorship



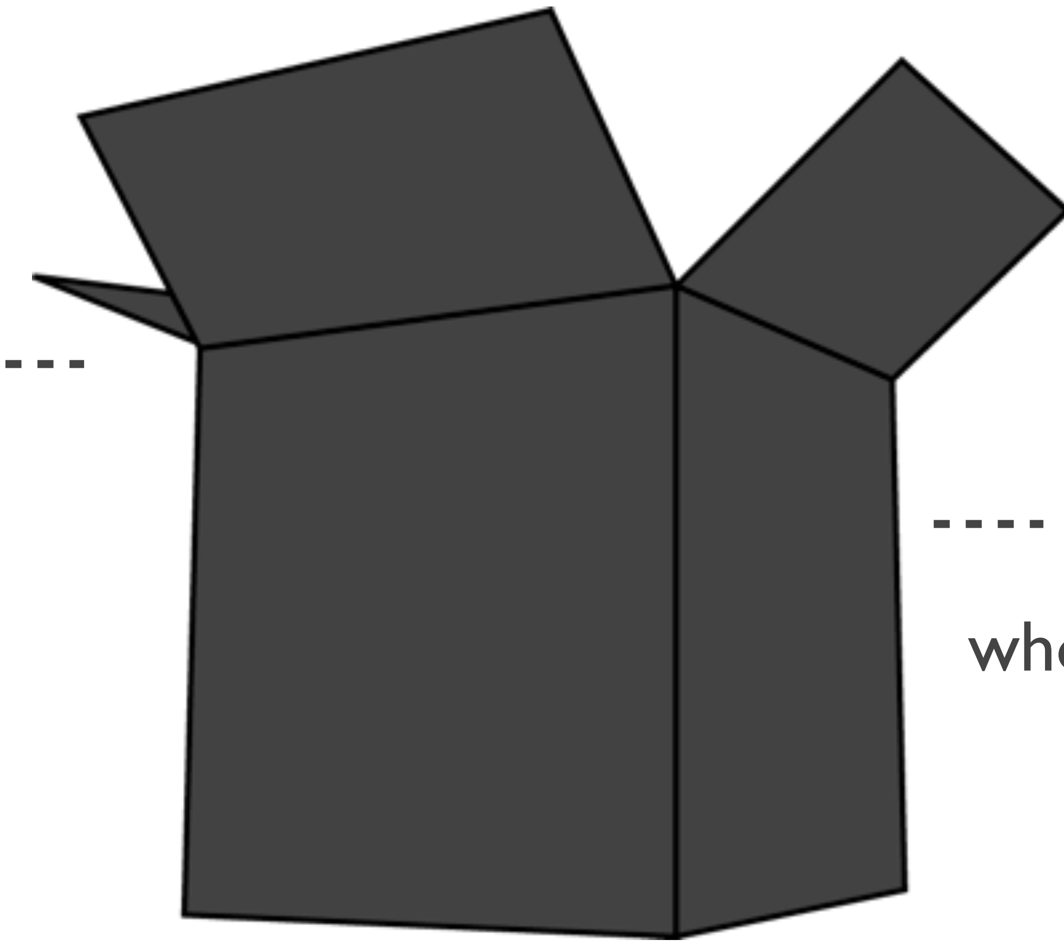
Compiler

family

version

optimization level

source language



Link & post-link

whole-program optimization

rewriting tools

obfuscation tools

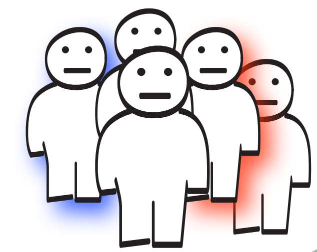
System

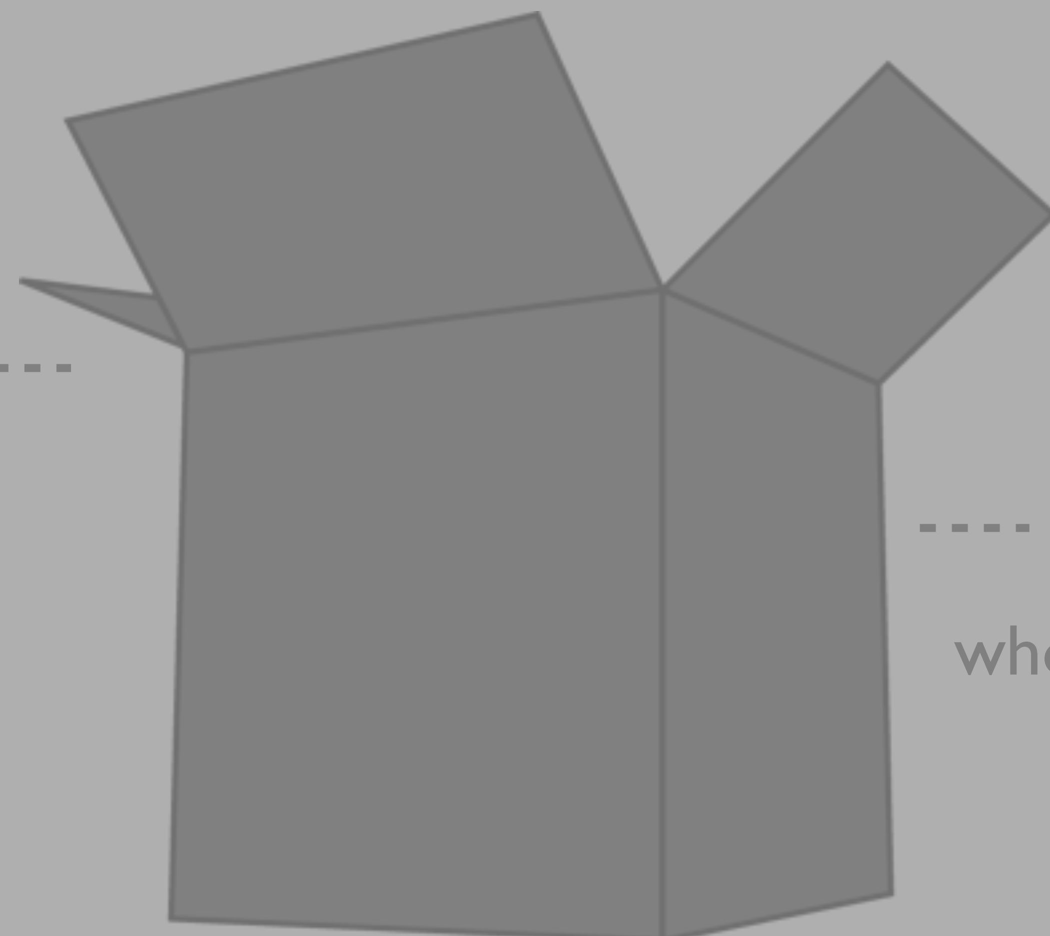
glibc static code

library imports

Program provenance

Authorship





Compiler

family

version

optimization level

source language

Link & post-link

whole-program optimization

rewriting tools

obfuscation tools

System

glibc static code

library imports

Applications

Debugging remote deployments

Applications

Debugging remote deployments

Applications

Debugging remote deployments

compiler bug?

subtle incompatibility?

Applications

Debugging remote deployments

compiler bug?

subtle incompatibility?

linux-vdso.so.1
libpthread.so.0
libasound.so.2
libdl.so.2
libstdc++.so.6
libm.so.6
libgcc_s.so.1
libc.so.6
/lib64/ld-linux-x86-64.so.2
librt.so.1

Applications

Debugging remote deployments

compiler bug?

subtle incompatibility?

linux-vdso.so.1
libpthread.so.0
libasound.so.2
libdl.so.2
libstdc++.so.6
libm.so.6
libgcc_s.so.1
libc.so.6
/lib64/ld-linux-x86-64.so.2
librt.so.1

Forensics

Applications

Debugging remote deployments

compiler bug?

subtle incompatibility?

linux-vdso.so.1
libpthread.so.0
libasound.so.2
libdl.so.2
libstdc++.so.6
libm.so.6
libgcc_s.so.1
libc.so.6
/lib64/ld-linux-x86-64.so.2
librt.so.1

Forensics

reverse engineering

decompiling

tools, obfuscations?

Applications

Debugging remote deployments

compiler bug?

subtle incompatibility?

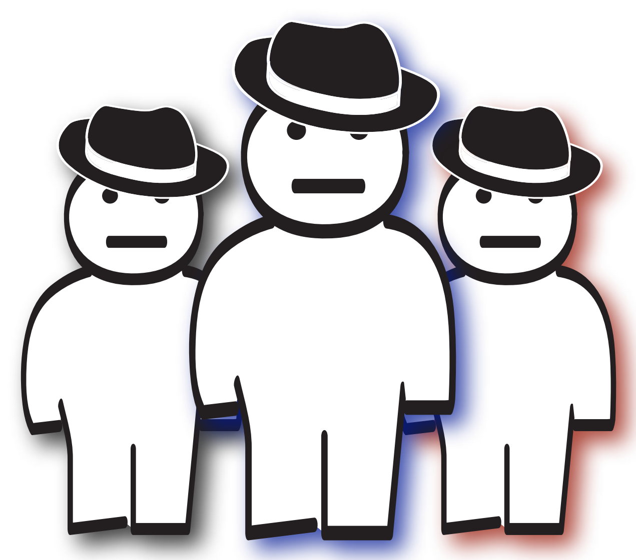
linux-vdso.so.1
libpthread.so.0
libasound.so.2
libdl.so.2
libstdc++.so.6
libm.so.6
libgcc_s.so.1
libc.so.6
/lib64/ld-linux-x86-64.so.2
librt.so.1

Forensics

reverse engineering

decompiling

tools, obfuscations?



Outline

Program Modeling

Compiler Toolchain

Authorship Attribution

Future Directions

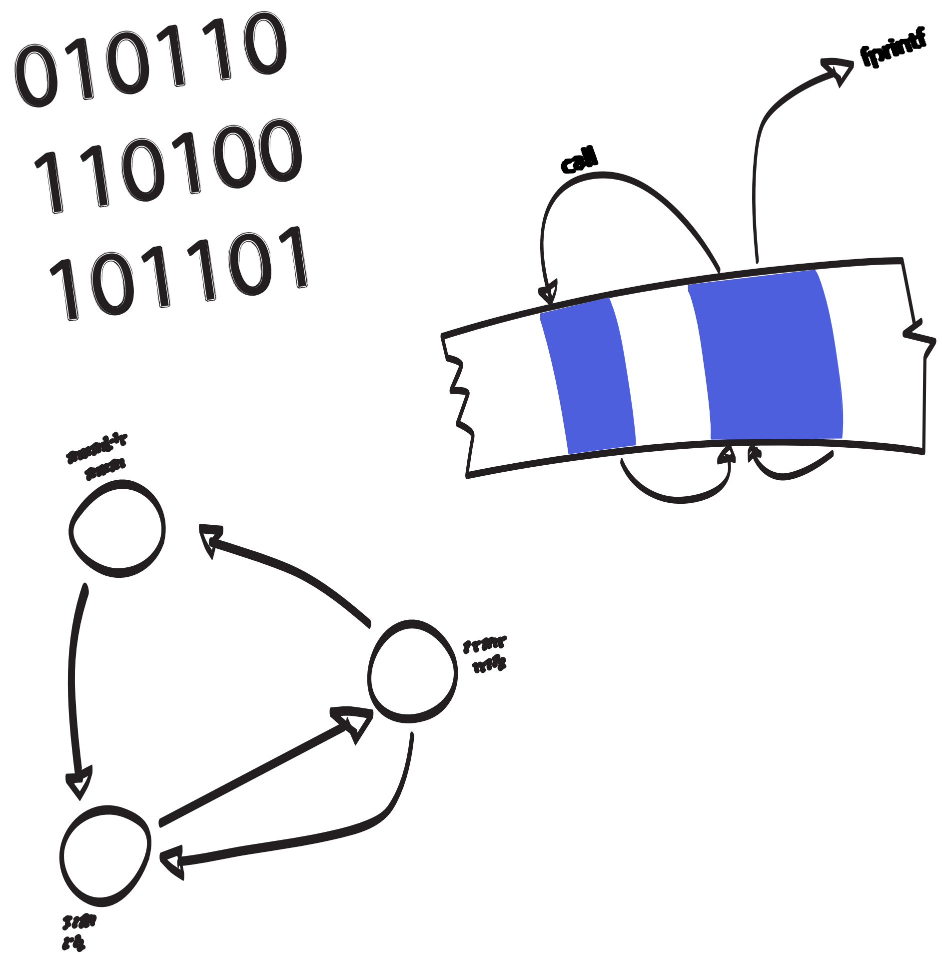
Outline

Program Modeling

Compiler Toolchain

Authorship Attribution

Future Directions



Digression: finding code

program binary

7

Digression: finding code

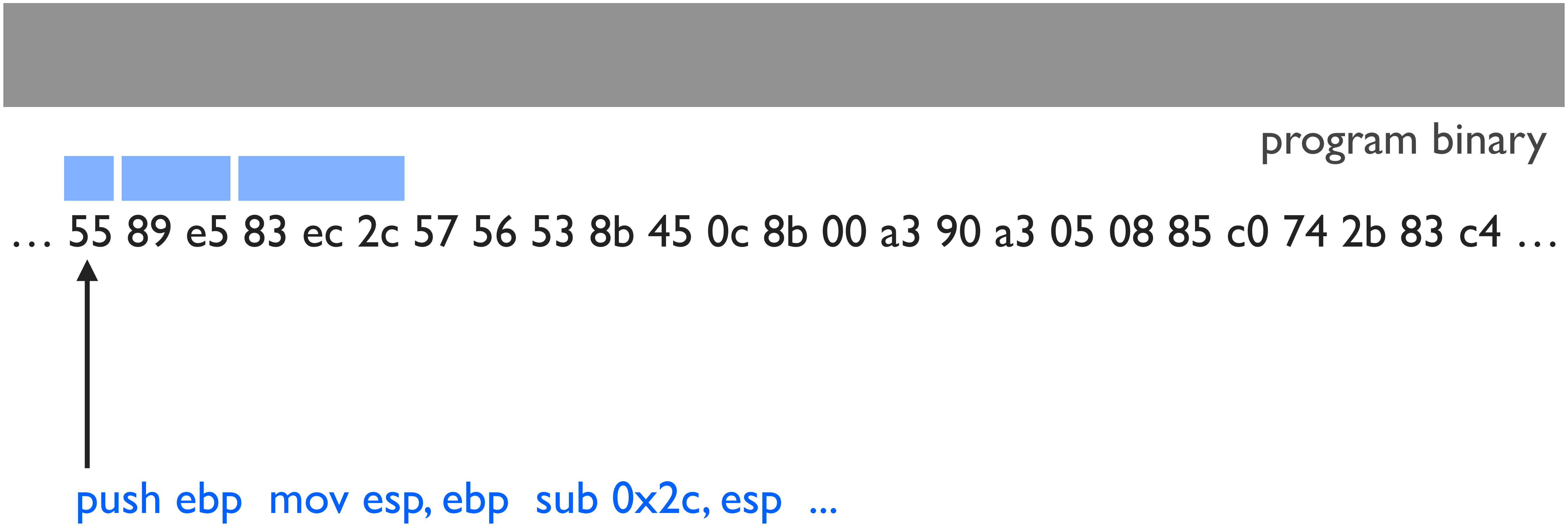
Digression: finding code

Digression: finding code

program binary

… 55 89 e5 83 ec 2c 57 56 53 8b 45 0c 8b 00 a3 90 a3 05 08 85 c0 74 2b 83 c4 …

Digression: finding code



Digression: finding code

program binary

… 55 89 e5 83 ec 2c 57 56 53 8b 45 0c 8b 00 a3 90 a3 05 08 85 c0 74 2b 83 c4 …

in 0x83,eax in [dx],al sub 0x57,al ...

push ebp mov esp, ebp sub 0x2c, esp ...

Digression: finding code

program binary

… 55 89 e5 83 ec 2c 57 56 53 8b 45 0c 8b 00 a3 90 a3 05 08 85 c0 74 2b 83 c4 …

in 0x83,eax in [dx],al sub 0x57,al ...

push ebp mov esp, ebp sub 0x2c, esp ...

[AAAI 2008]

Model compiler-specific "function entry points"

Compute max-likelihood labels $P(\text{addr is FEP}|\ldots,\lambda) \propto \cdots$

F_1 from .86 - .99 *depending on compiler*

Instruction-level features

IDIOMS

`‹push EBP ; * ; mov ESP, EBP›`

`‹mov [IMM], RAX ; sub [IMM], RAX›`

Paradyn™

Instruction-level features

single-instruction wildcard

IDIOMS

```
‹push EBP ; * ; mov ESP, EBP›

‹mov [IMM], RAX ; sub [IMM], RAX›
```

opcode class abstraction

hidden immediates

Instruction-level features

single-instruction wildcard

IDIOMS

```
<push EBP ; * ; mov ESP, EBP>
<mov [IMM], RAX ; sub [IMM], RAX>
```

opcode class abstraction

hidden immediates

N-GRAMS

```
<4889c2be>  <018b45f8>  <8d45f8>
```

4-grams

3-grams

Binary code model

… 55 89 e5 83 ec 2c 57 56 53 8b 45 0c 8b 00 a3 90 a3 05 08 85 c0 74 2b 83 c4 …

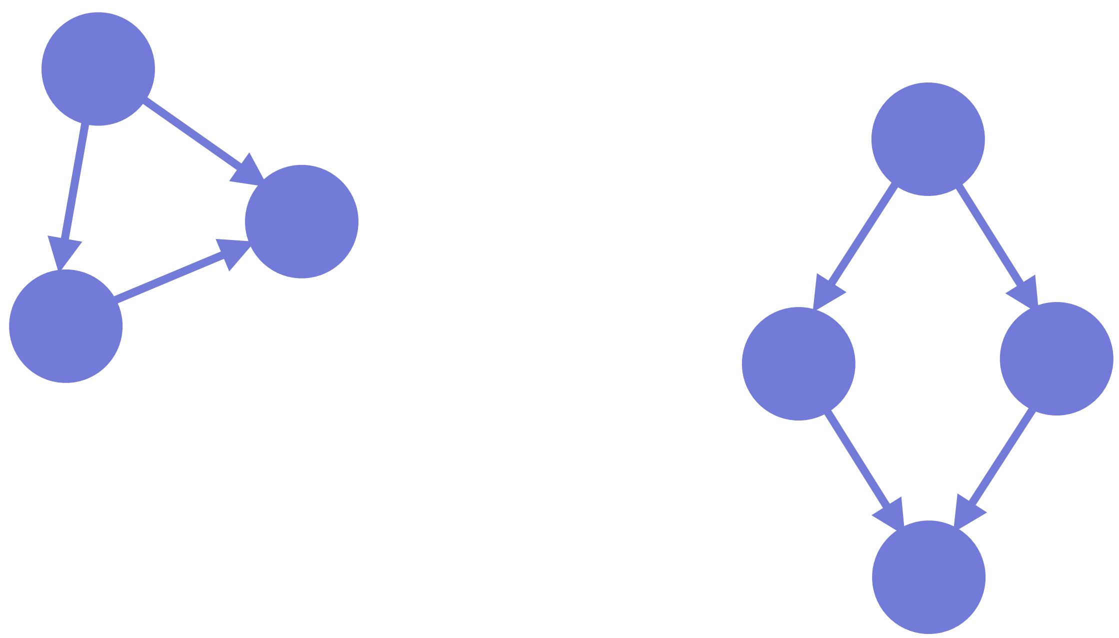
`<push EBP ; * ; mov ESP, EBP>` `<mov [IMM], RAX ; sub [IMM], RAX>`

Binary code model

program binary

… 55 89 e5 83 ec 2c 57 56 53 8b 45 0c 8b 00 a3 90 a3 05 08 85 c0 74 2b 83 c4 …

`<push EBP ; * ; mov ESP, EBP>` `<mov [IMM], RAX ; sub [IMM], RAX>`



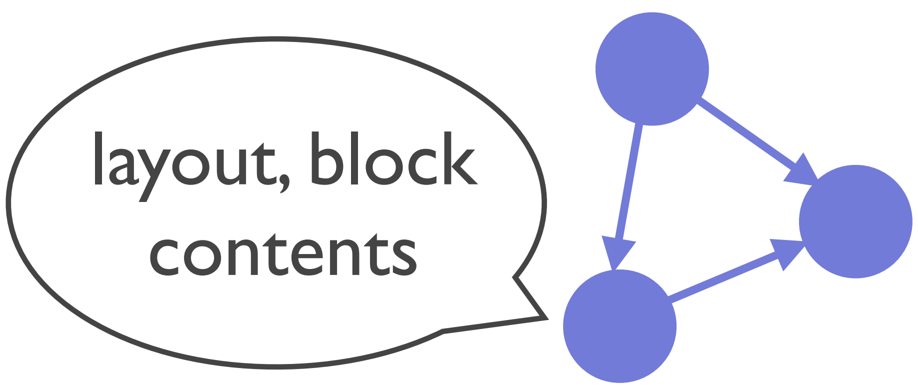
Control Flow Graph

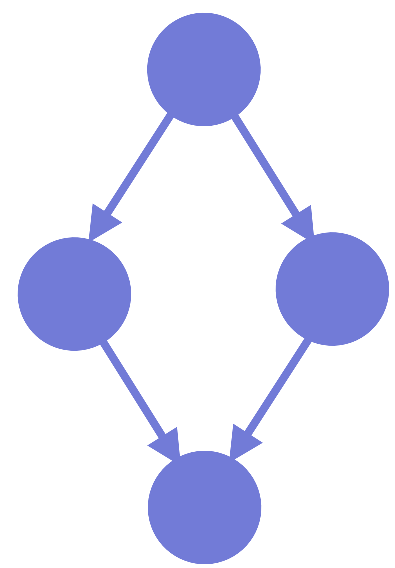
Binary code model

program binary

… 55 89 e5 83 ec 2c 57 56 53 8b 45 0c 8b 00 a3 90 a3 05 08 85 c0 74 2b 83 c4 …

<push EBP ; * ; mov ESP, EBP>   <mov [IMM], RAX ; sub [IMM], RAX>



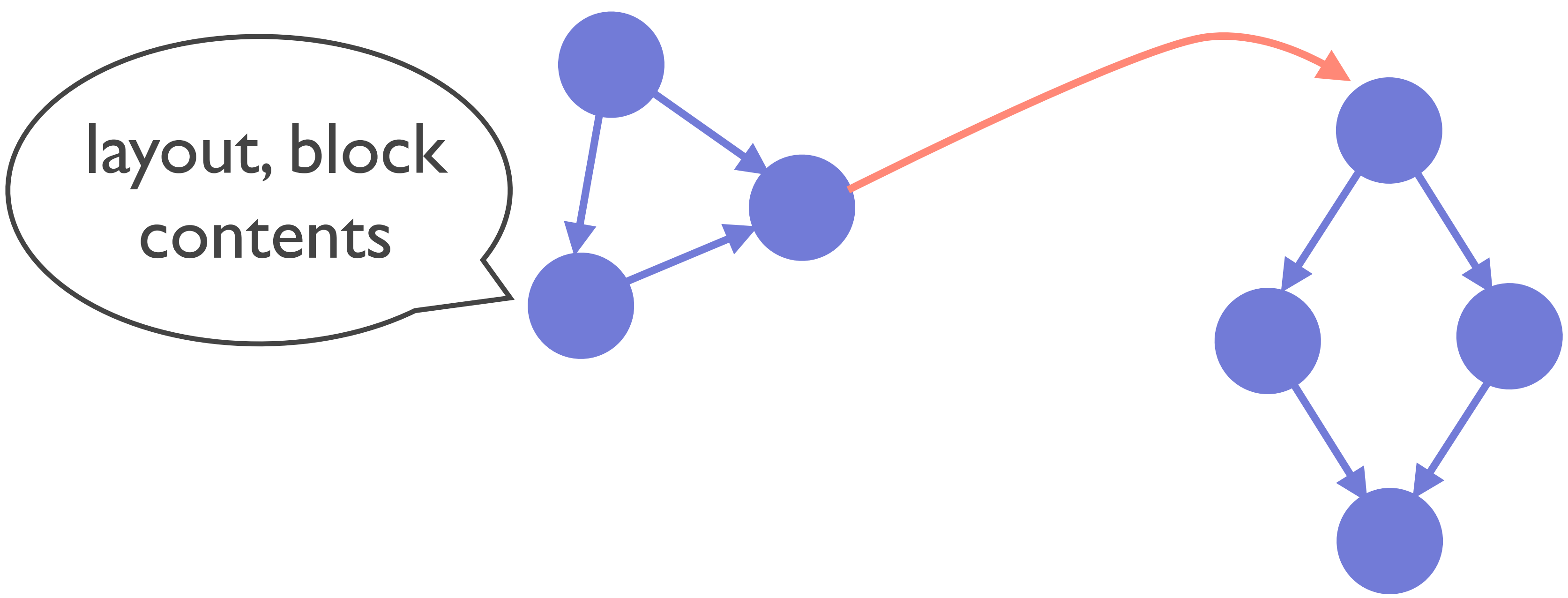


Control Flow Graph

Binary code model

… 55 89 e5 83 ec 2c 57 56 53 8b 45 0c 8b 00 a3 90 a3 05 08 85 c0 74 2b 83 c4 …

`<push EBP ; * ; mov ESP, EBP>` `<mov [IMM], RAX ; sub [IMM], RAX>`



Control Flow Graph

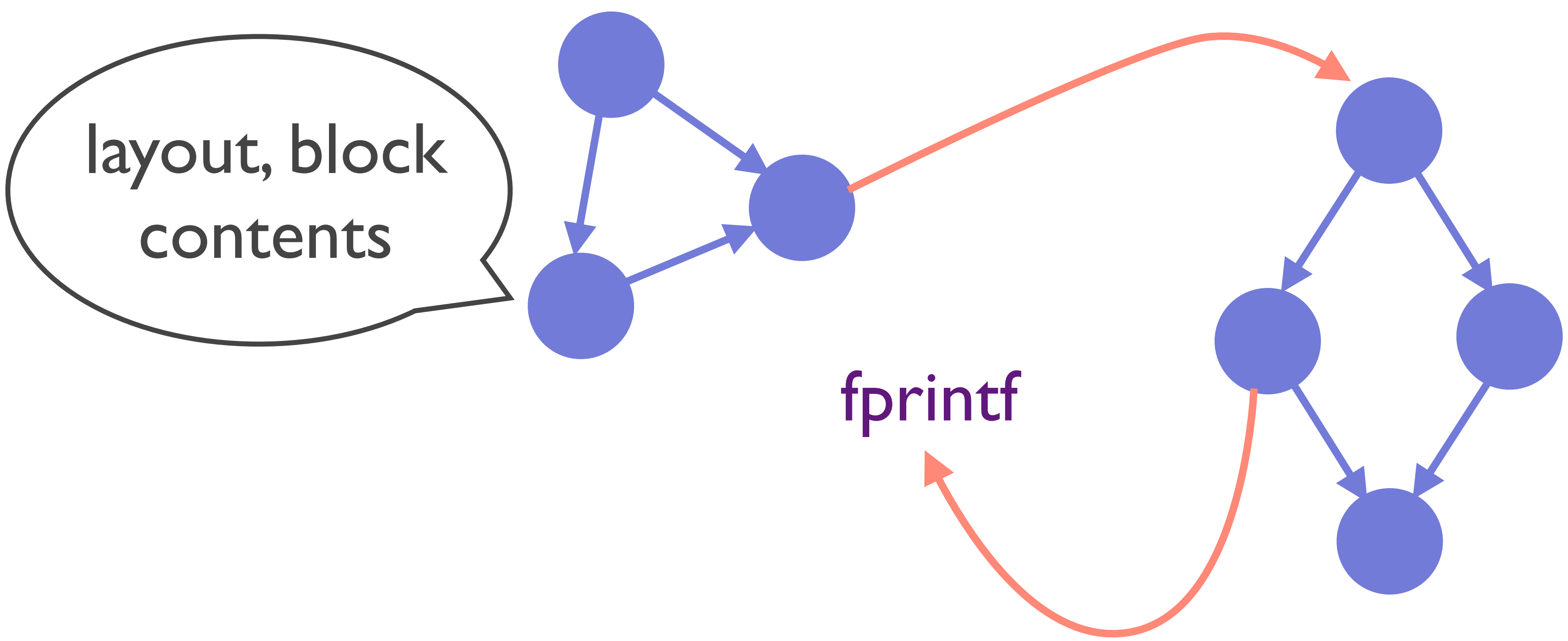
Call Graph

Binary code model

program binary

… 55 89 e5 83 ec 2c 57 56 53 8b 45 0c 8b 00 a3 90 a3 05 08 85 c0 74 2b 83 c4 …

<push EBP ; * ; mov ESP, EBP> <mov [IMM], RAX ; sub [IMM], RAX>

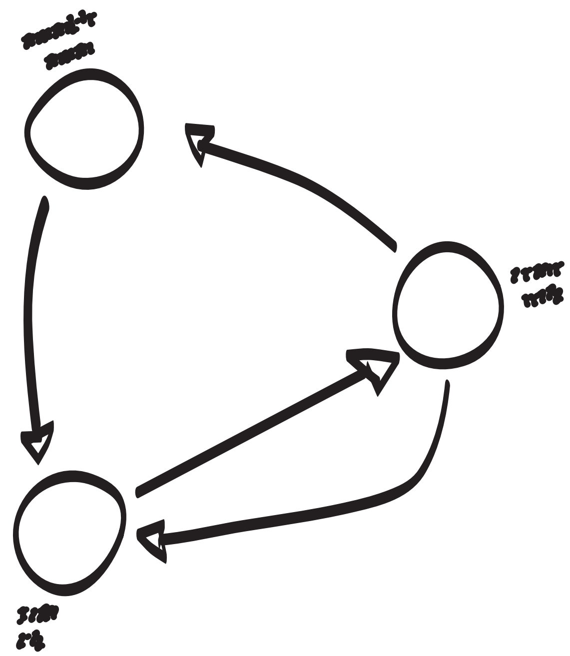


Control Flow Graph

Call Graph

External Libraries

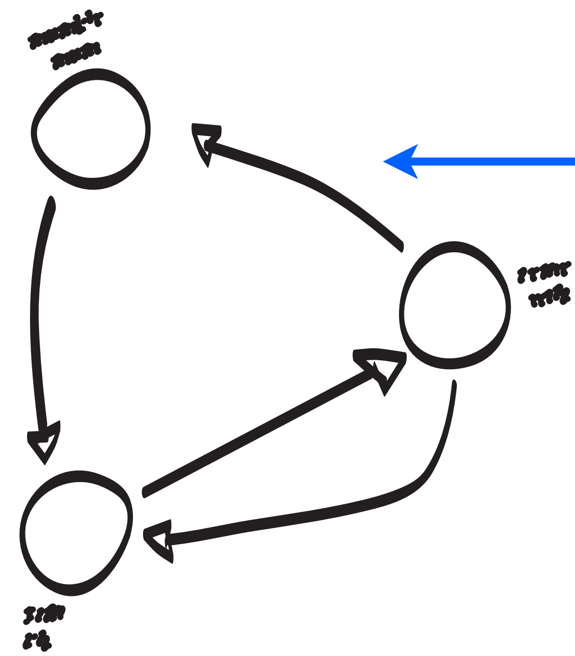
Graphlets



Graphlets

code element nodes

(e.g. basic blocks)

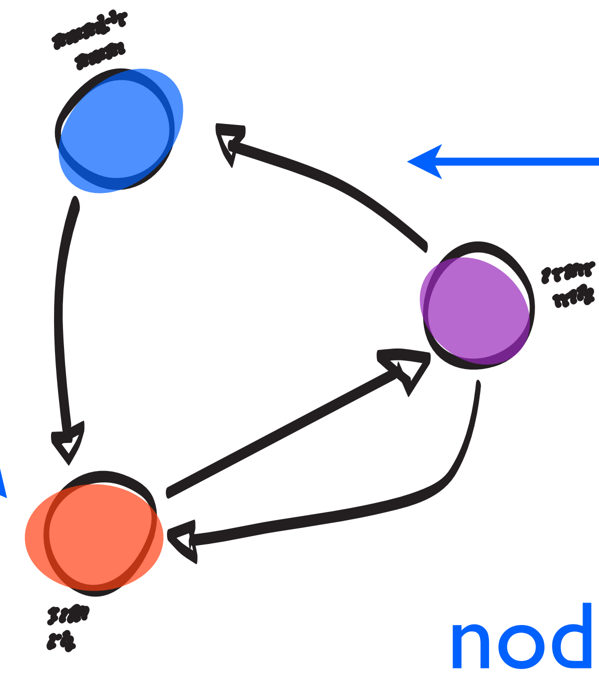


typed edges

(branch, call, etc.)

Graphlets

code element nodes
(e.g. basic blocks)

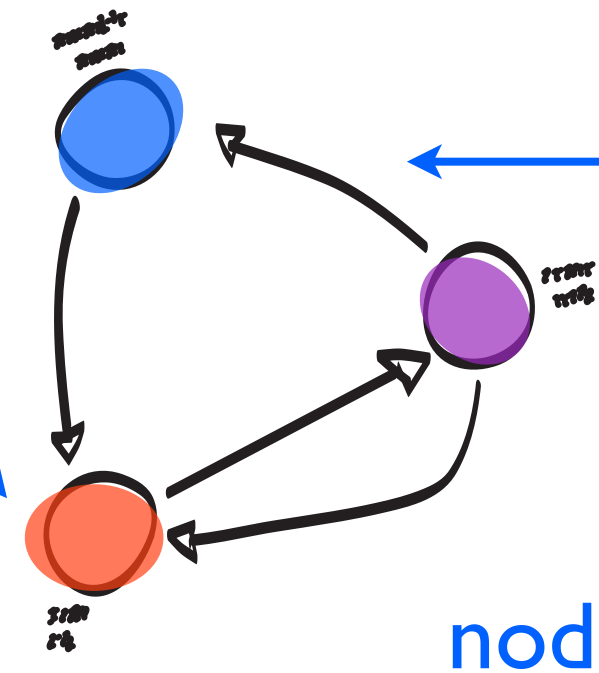


typed edges
(branch, call, etc.)

node colors

Graphlets

code element nodes
(e.g. basic blocks)

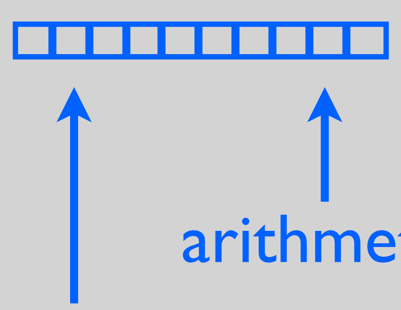


typed edges
(branch, call, etc.)

node colors

Ex: instruction summary graphlets

Color bit field

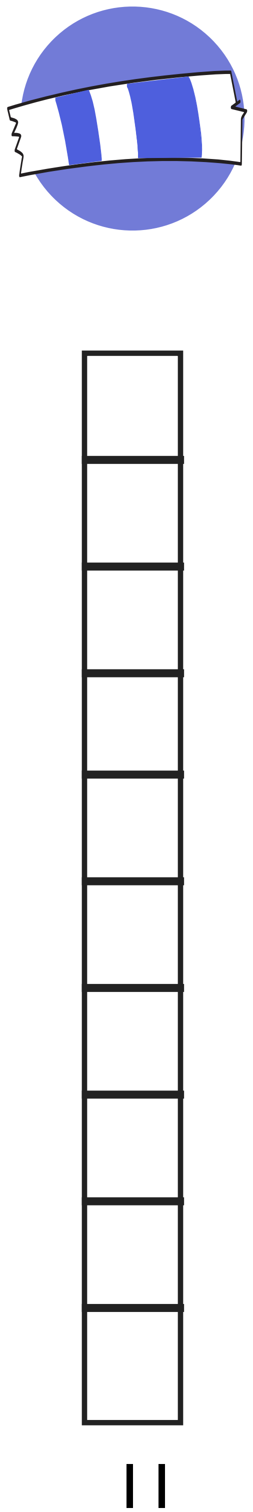


14 instruction categories

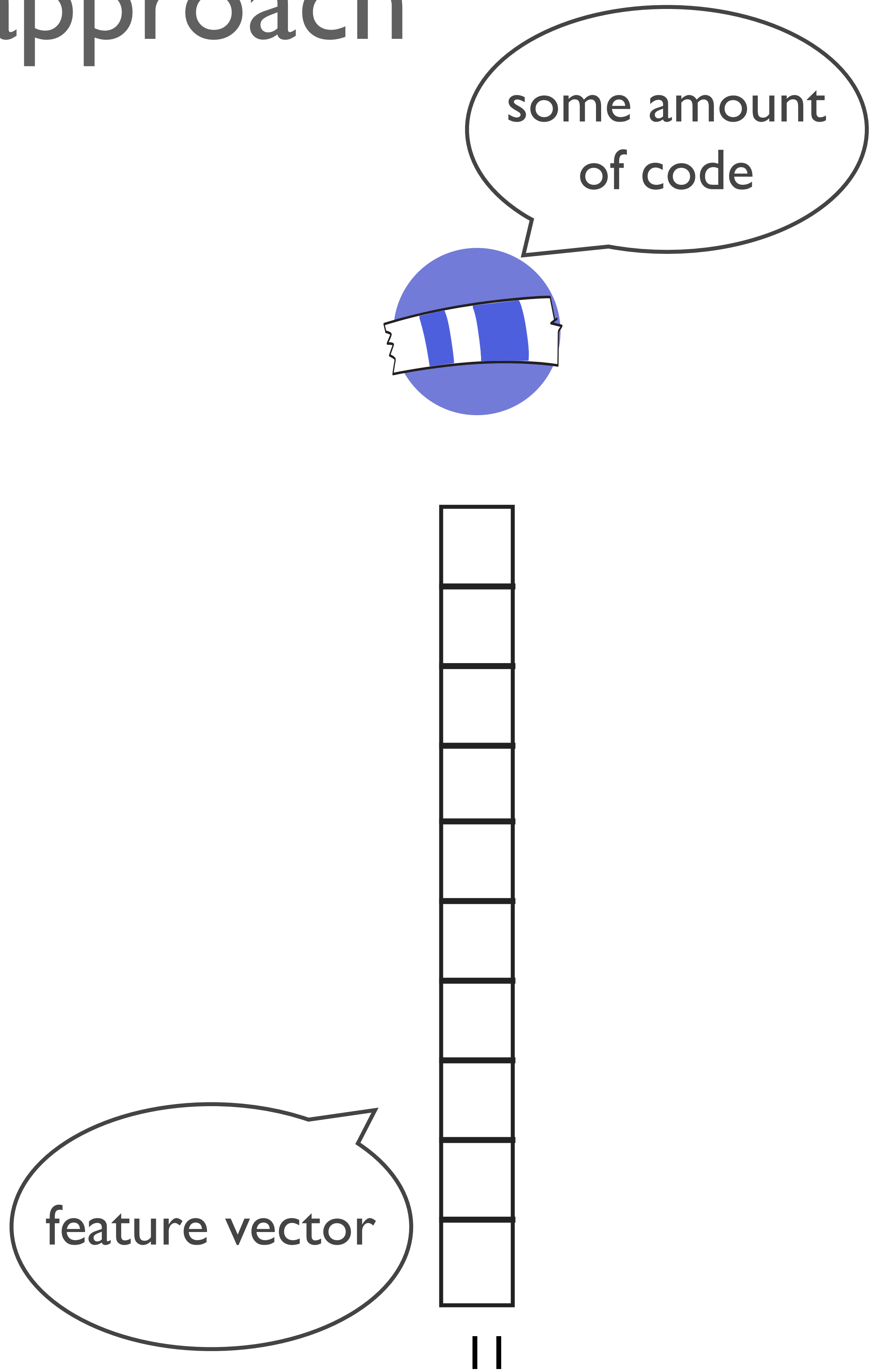
2^{14} possible colors

sparse in practice

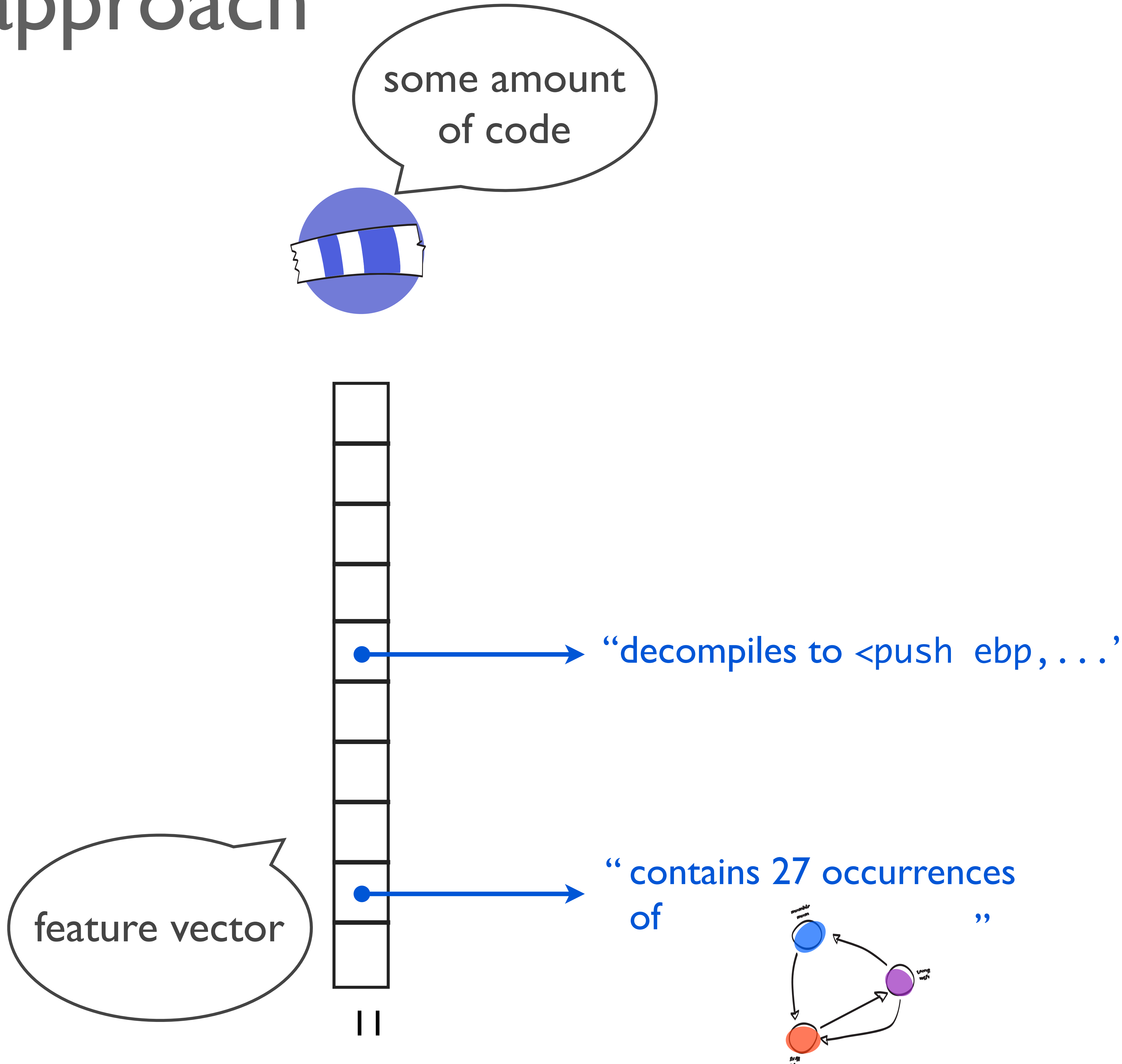
Modeling approach



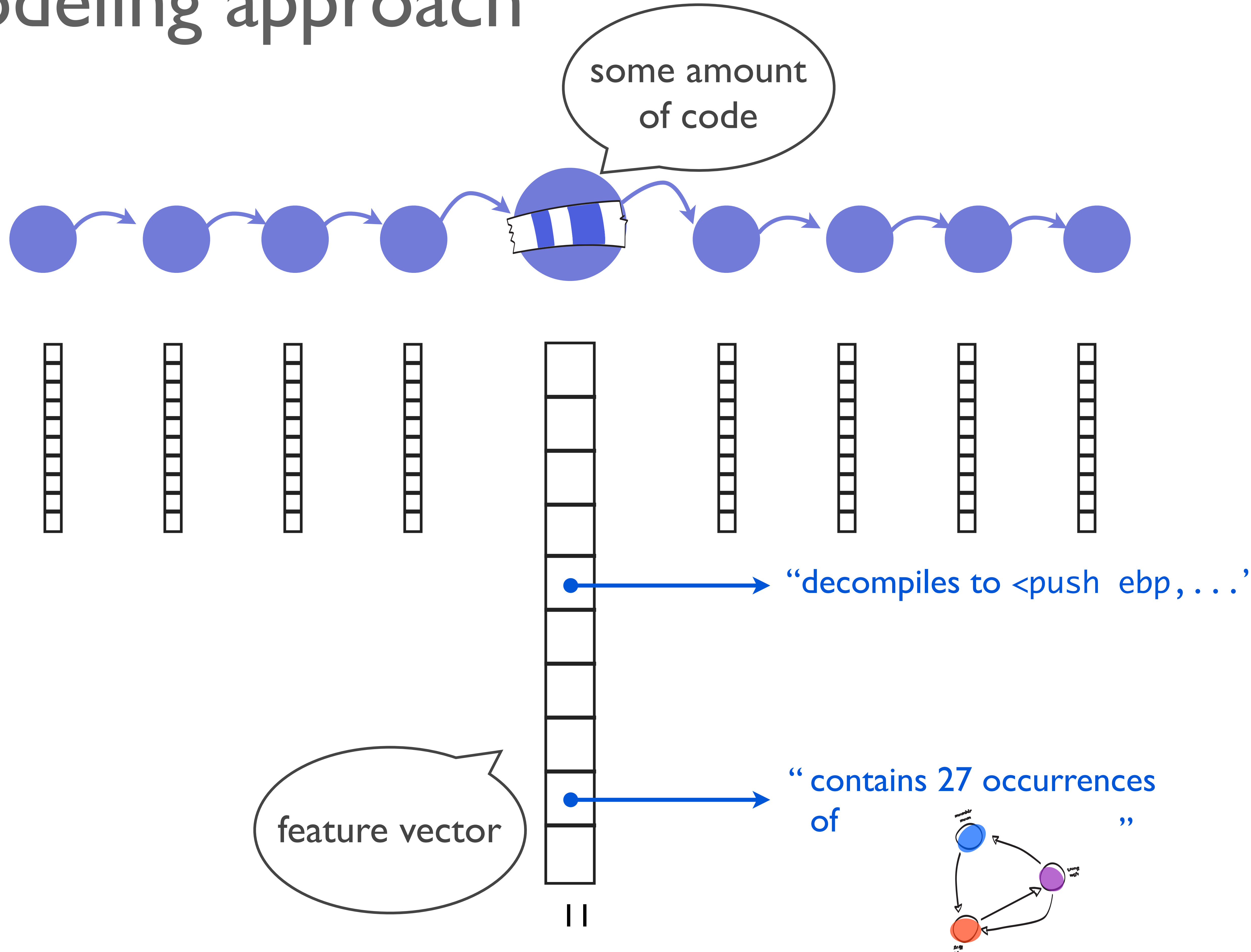
Modeling approach



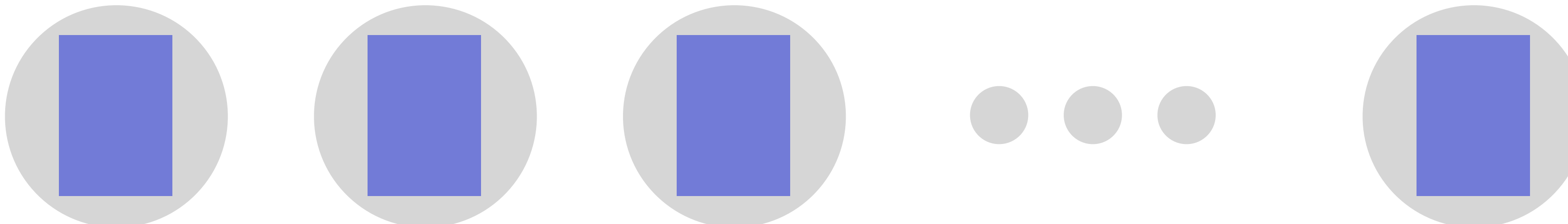
Modeling approach



Modeling approach

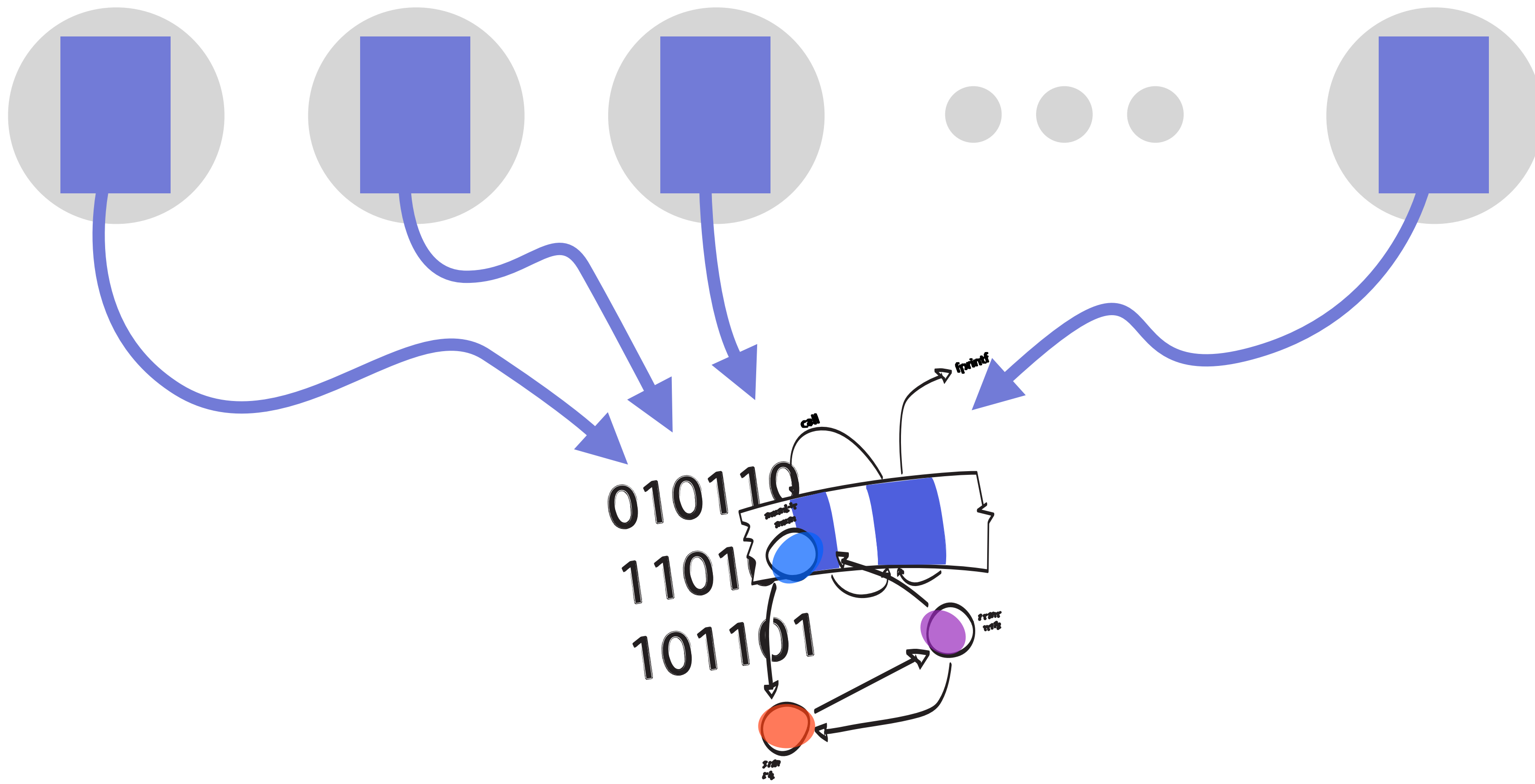


Feature selection



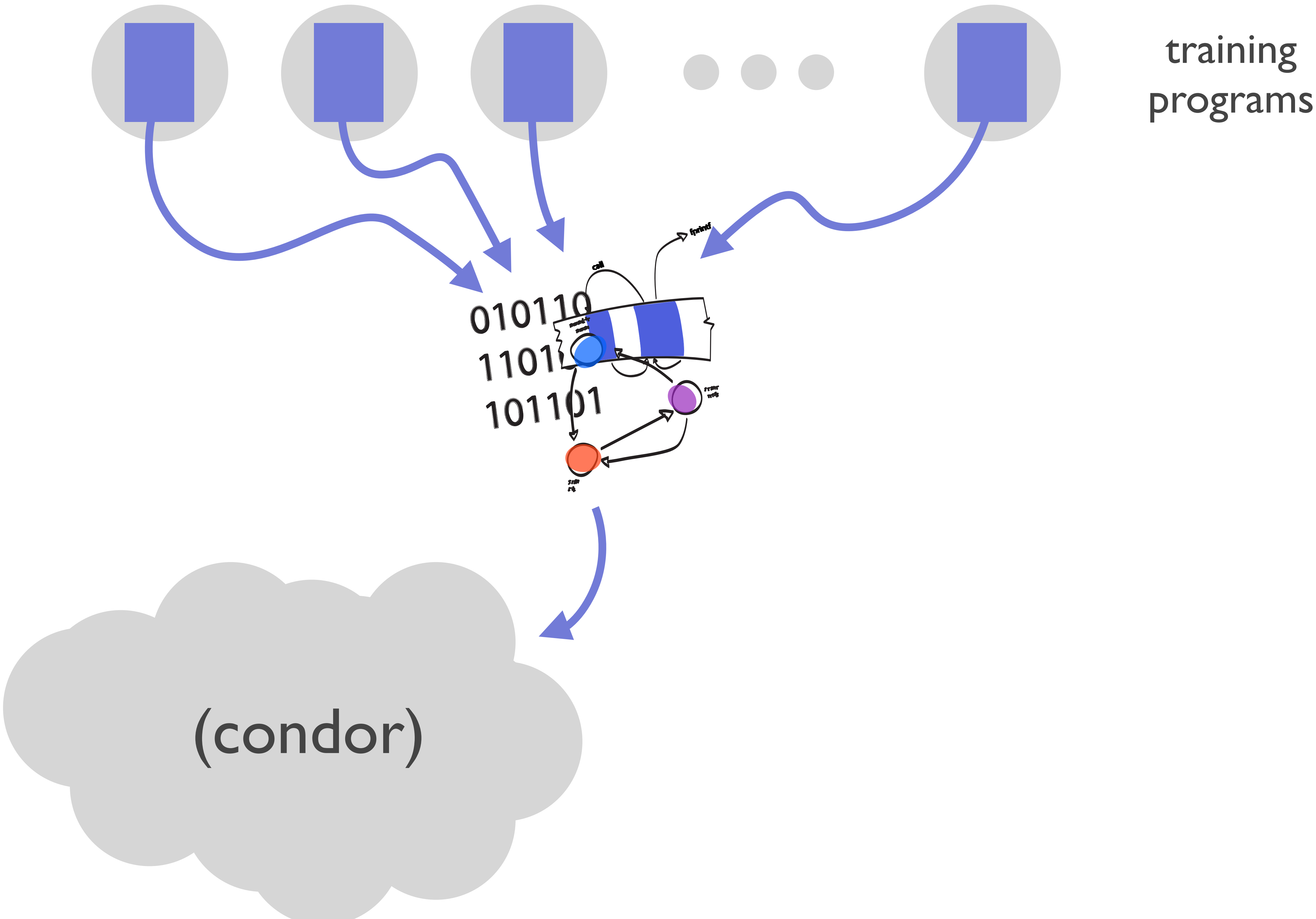
training programs

Feature selection



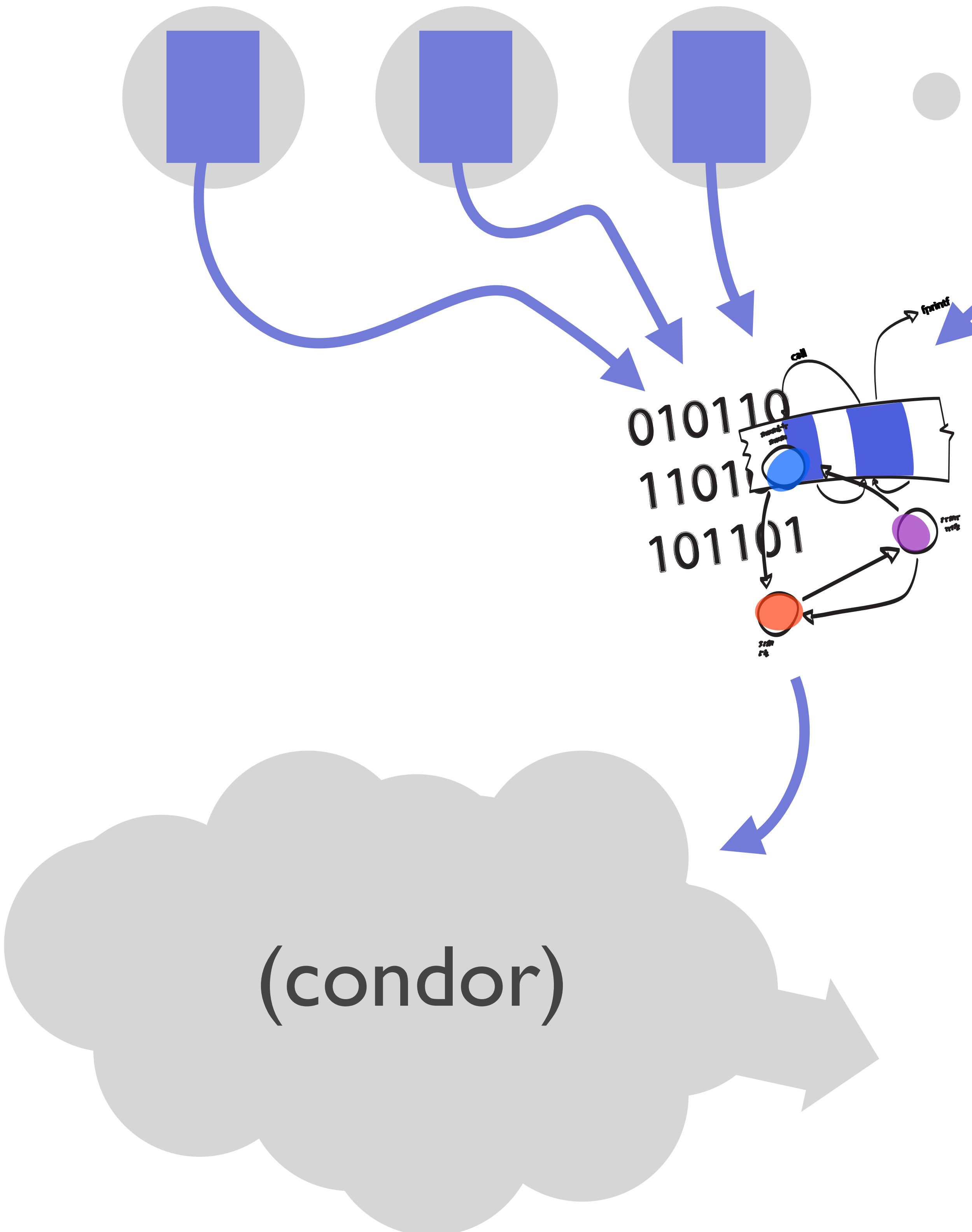
training programs

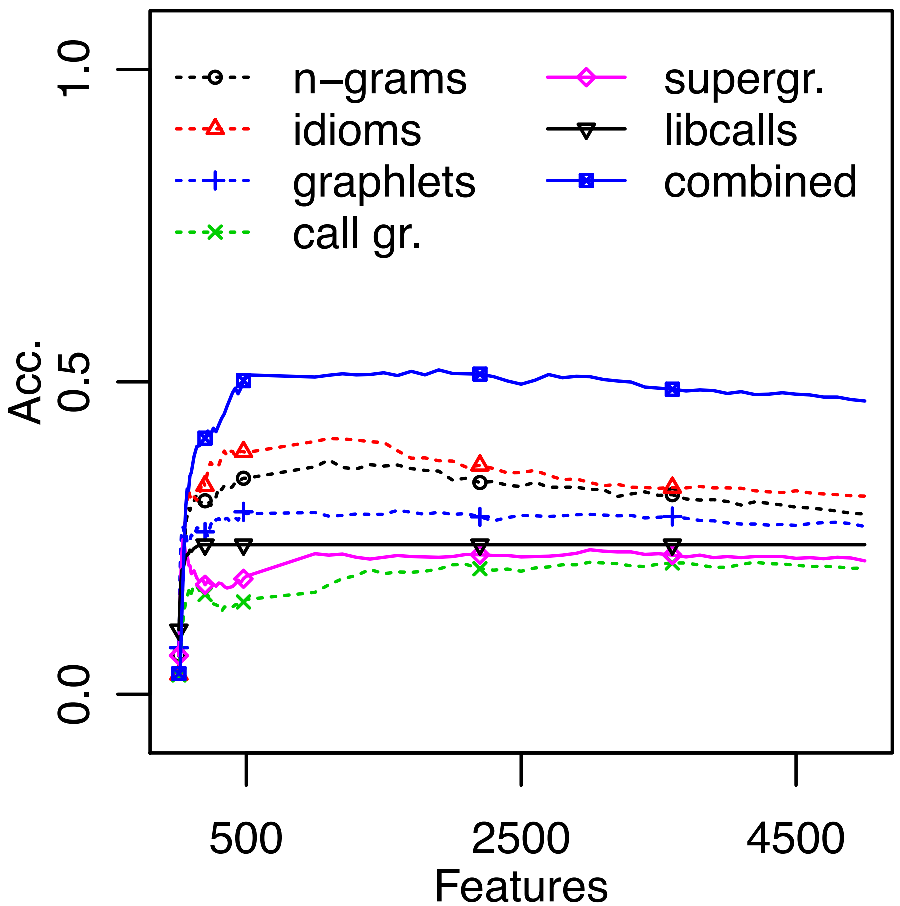
Feature selection



Feature selection

training programs





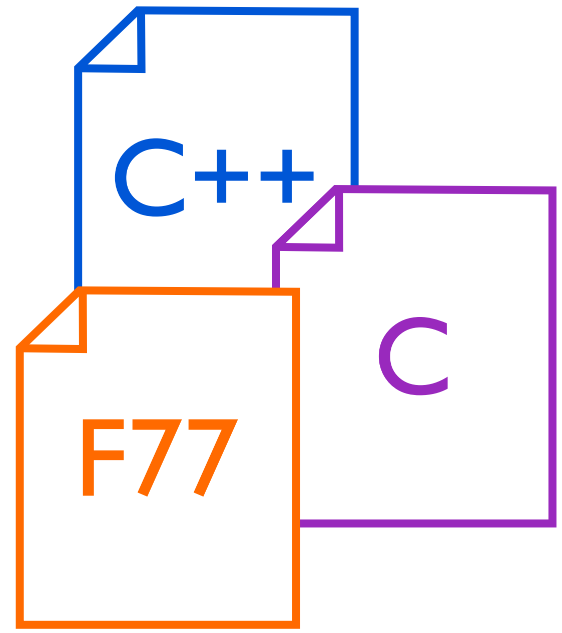
Outline

Program Modeling

Compiler Toolchain

Authorship Attribution

Future Directions



Compiler toolchain

C

C++

F77

GCC

Microsoft Visual Studio

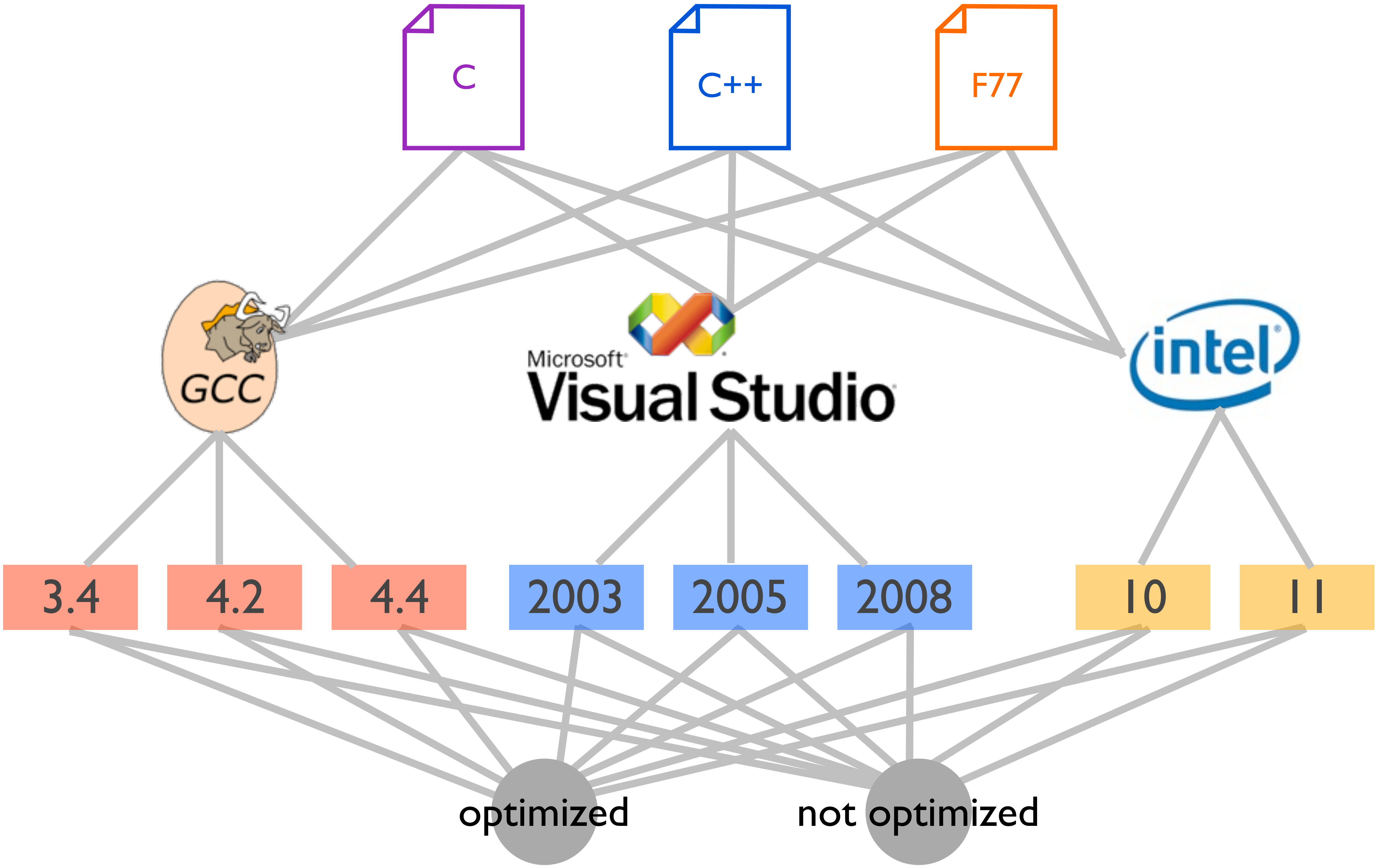
intel

3.4 | 4.2 | 4.4 | 2003 | 2005 | 2008 | 10 | 11

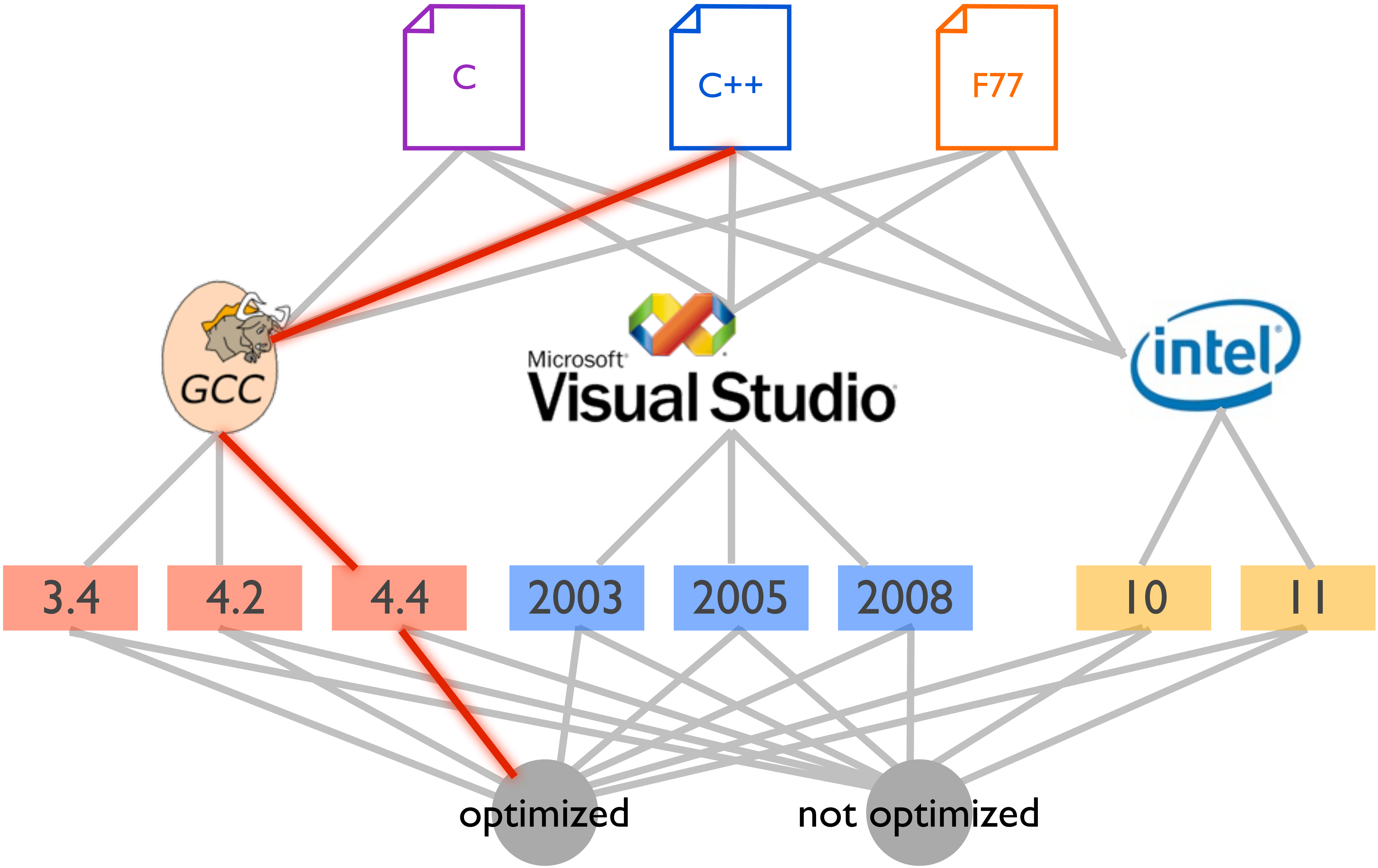
optimized

not optimized

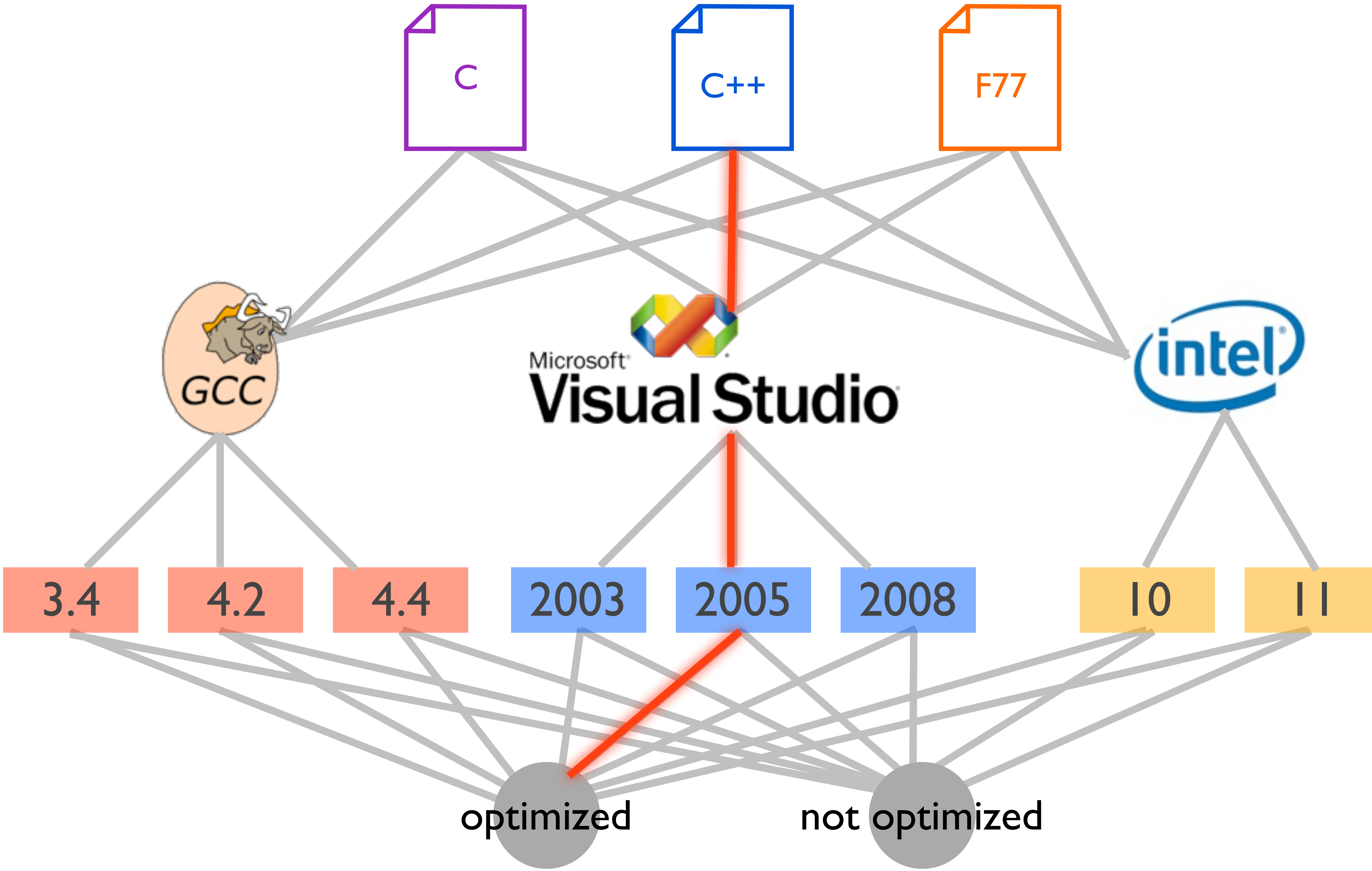
Compiler toolchain



Compiler toolchain



Compiler toolchain



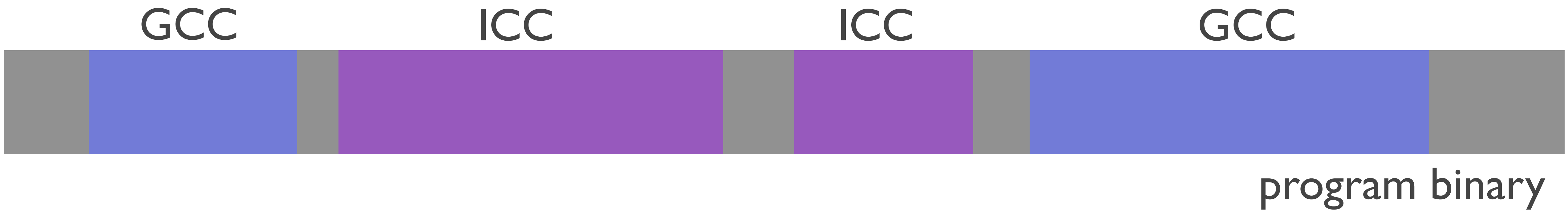
Byte-sequence model

[PASTE 2010]

program binary

Byte-sequence model

[PASTE 2010]



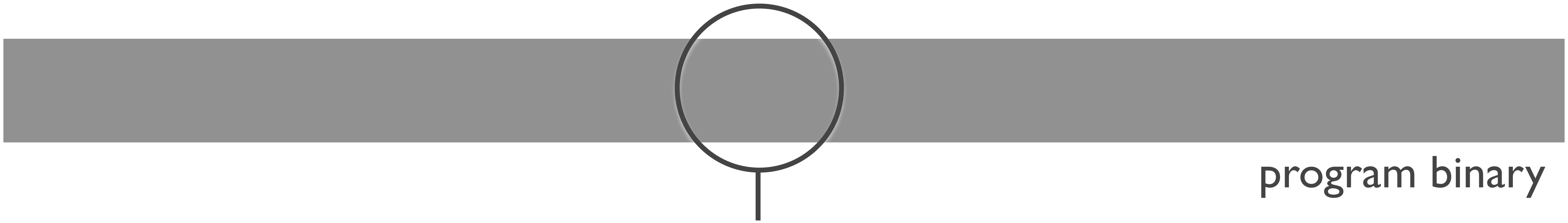
Byte-sequence model

[PASTE 2010]

program binary

Byte-sequence model

[PASTE 2010]

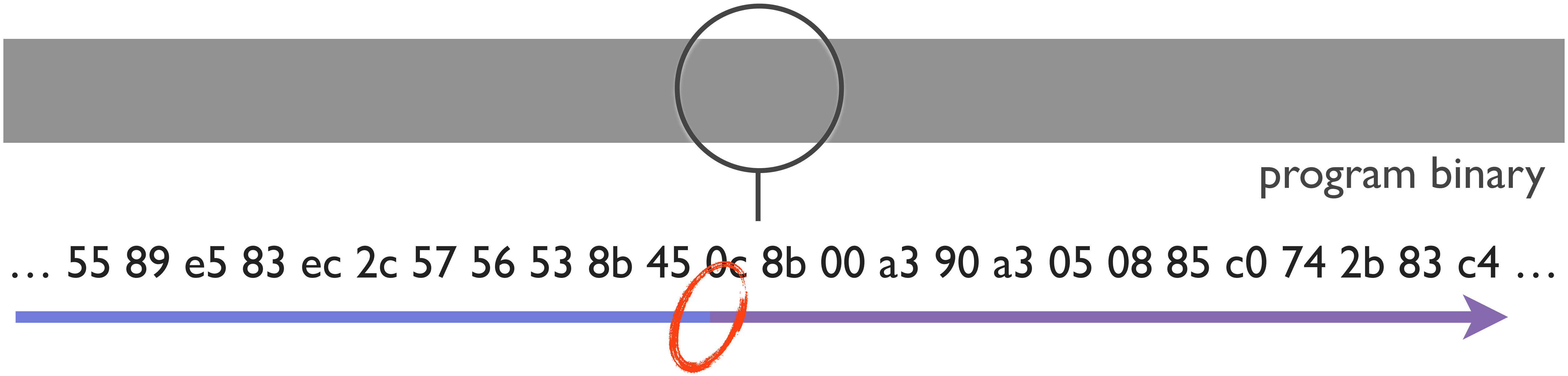


program binary

… 55 89 e5 83 ec 2c 57 56 53 8b 45 0c 8b 00 a3 90 a3 05 08 85 c0 74 2b 83 c4 …

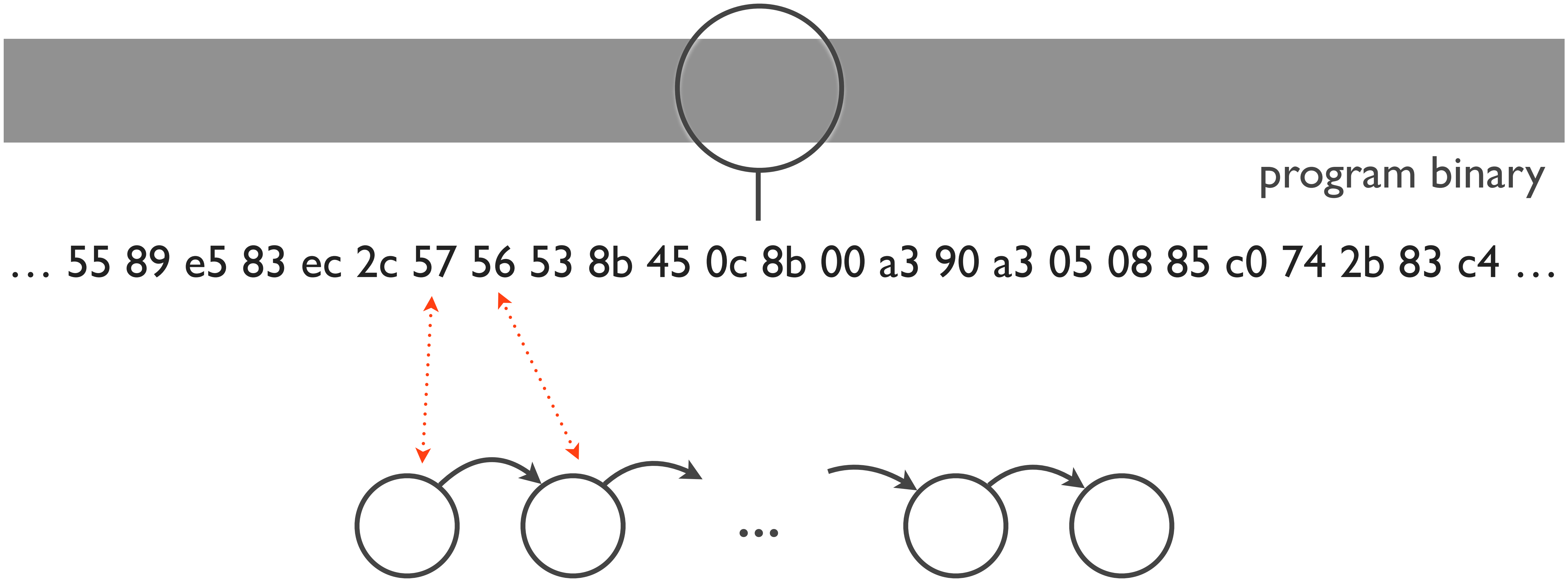
Byte-sequence model

[PASTE 2010]



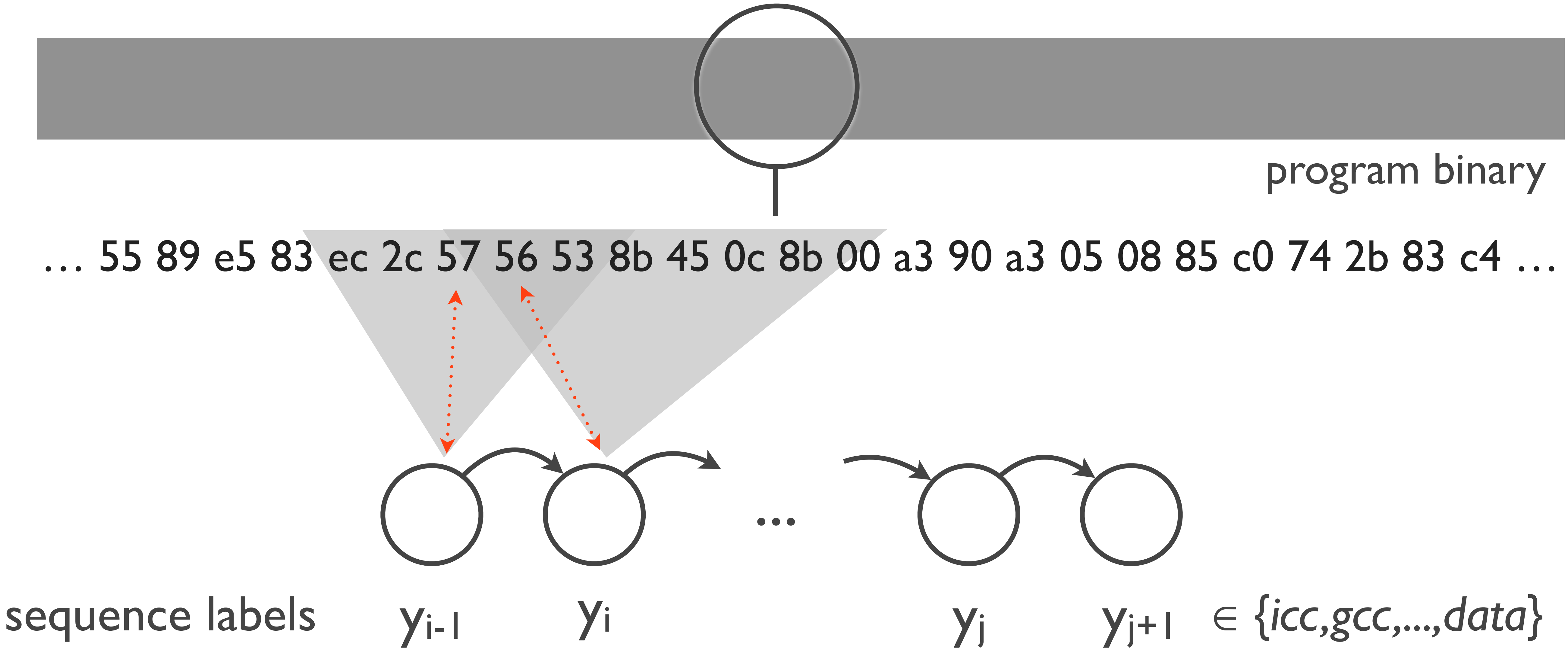
Byte-sequence model

[PASTE 2010]



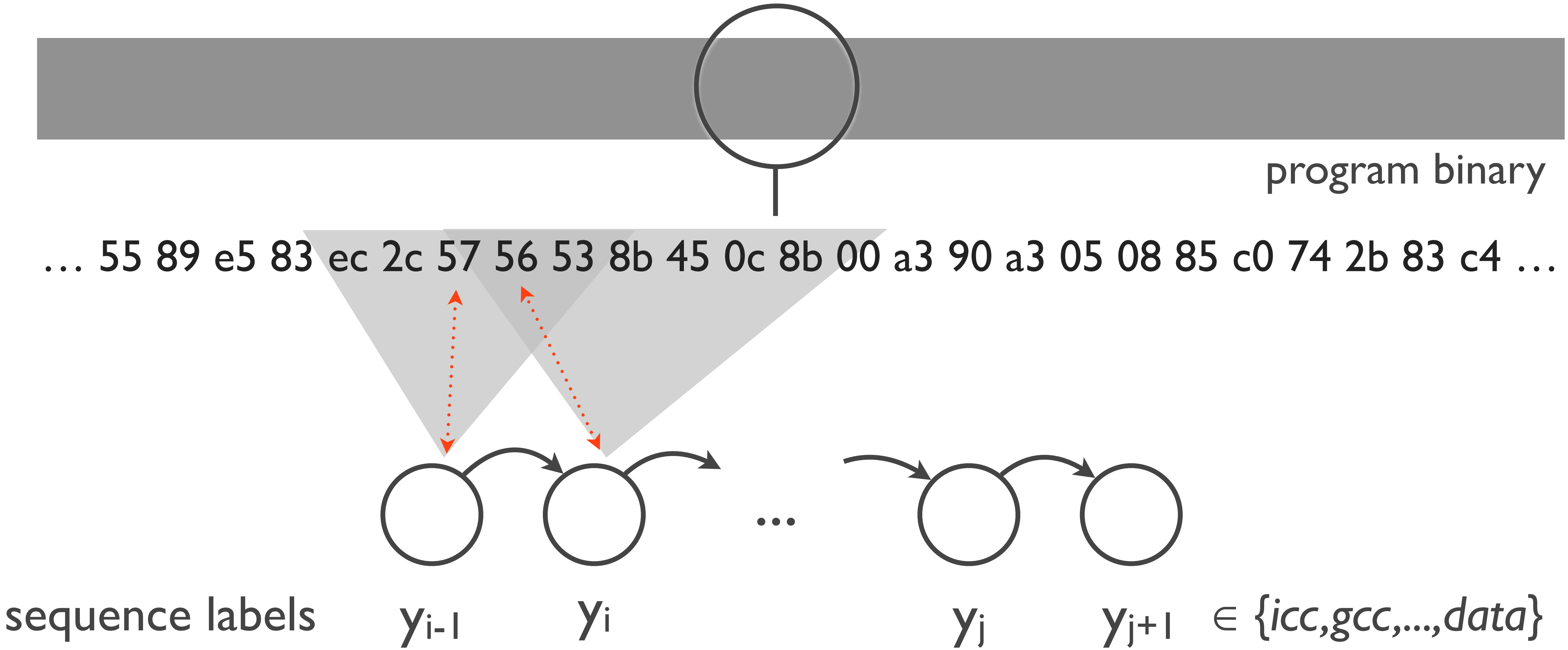
Byte-sequence model

[PASTE 2010]



Byte-sequence model

[PASTE 2010]



Compiler labels modeled as CRF...

Digression: Conditional Random Fields

$$P(Y|X) \propto \exp\left(\sum_{i=1}^{K}\sum_{u\in\mathcal{U}}\lambda_{y_i,u}\cdot f_u(x_i)\right)$$

Digression: Conditional Random Fields

$$\underbrace{P(\overbrace{Y}^{\text{labels}}|\underbrace{X}_{\text{evidence}})}_{} \propto \exp\left(\sum_{i=1}^{K}\sum_{u\in\mathcal{U}} \overbrace{\lambda_{y_i,u}}^{\text{weights (learned)}} \cdot \underbrace{f_u(x_i)}_{\text{feature functions}}\right)$$

Digression: Conditional Random Fields

labels

weights (learned)

$$P(Y|X) \propto \exp\left(\sum_{i=1}^{K}\sum_{u\in\mathcal{U}} \lambda_{y_i,u} \cdot f_u(x_i)\right)$$

evidence

feature functions

Idiom *feature function*

$$f_u = \begin{cases} 1 & \text{if } x_i \text{ decompiles to idiom } u \\ 0 & \text{otherwise} \end{cases}$$

Digression: Conditional Random Fields

labels

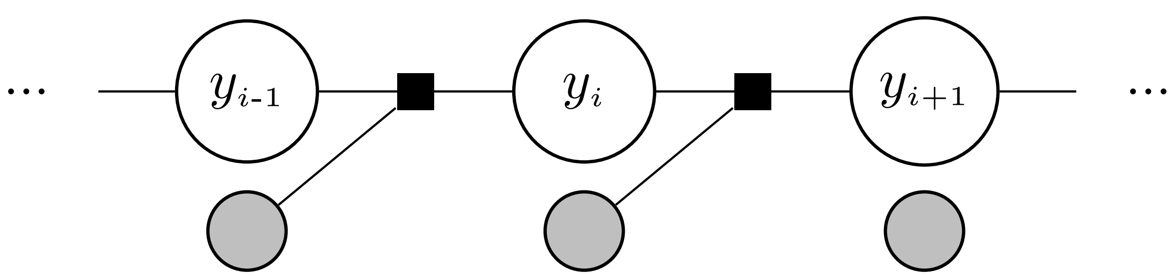
weights (learned)

$$P(Y|X) \propto \exp\left(\sum_{i=1}^{K}\sum_{u\in\mathcal{U}} \lambda_{y_i,u} \cdot f_u(x_i)\right)$$

evidence

feature functions

Linear chain CRF



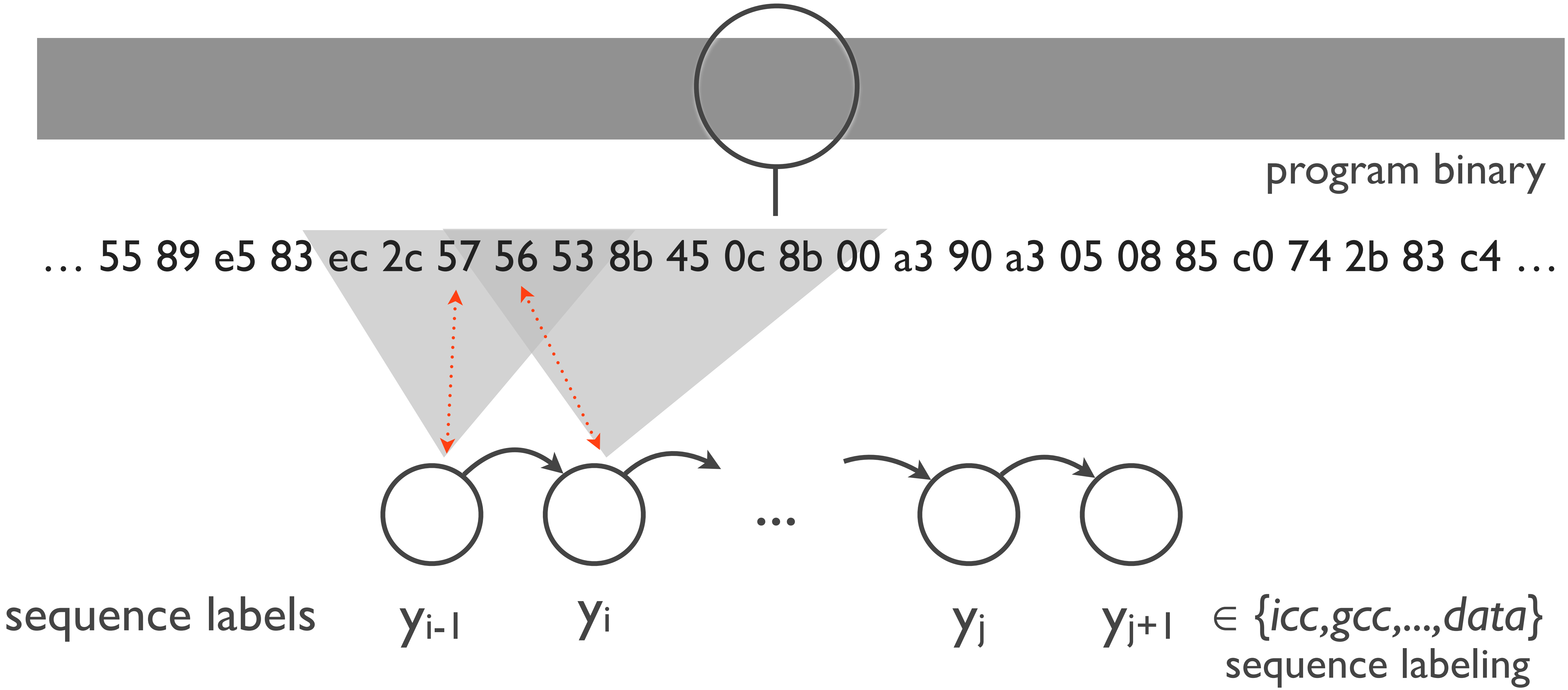
exact inference tractable

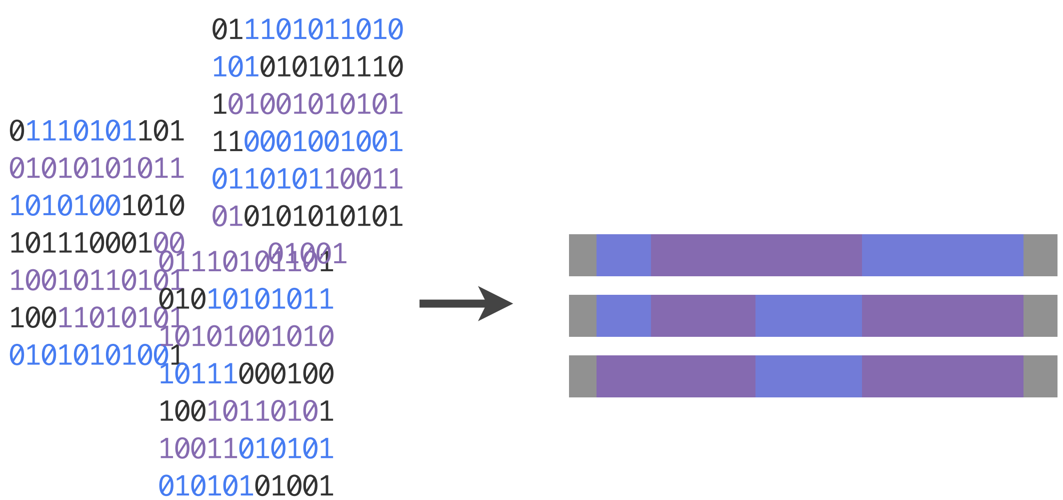
Idiom *feature function*

$$f_u = \begin{cases} 1 & \text{if } x_i \text{ decompiles to idiom } u \\ 0 & \text{otherwise} \end{cases}$$

Byte-sequence model

[PASTE 2010]





94% accuracy labeling mixed-compiler sequences

+18% accuracy increase in function finding

Toolchain details

[ISSTA 2011]

compiler family

GNU, Intel,
Microsoft

Toolchain details

[ISSTA 2011]

source language	compiler family	version	optimization level
C, C++, Fortran	GNU, Intel, Microsoft	[several]	low, high

Toolchain details

[ISSTA 2011]

source language	compiler family	version	optimization level
C, C++, Fortran	GNU, Intel, Microsoft	[several]	low, high

Toolchain details

[ISSTA 2011]

source language	compiler family	version	optimization level
C, C++, Fortran	GNU, Intel, Microsoft	[several]	low, high

functions

Toolchain details

[ISSTA 2011]

source language	compiler family	version	optimization level
C, C++, Fortran	GNU, Intel, Microsoft	[several]	low, high

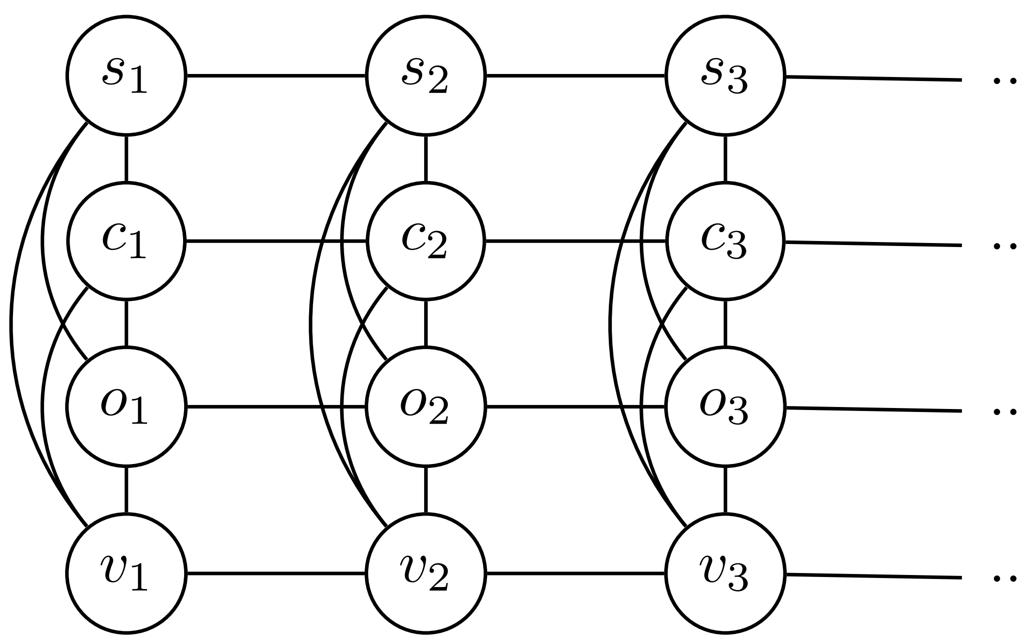
functions

language

family

optimization

version



idioms + graphlets

Paradyn

Toolchain details

[ISSTA 2011]

	Individually (SVM)	Linear CRF	Joint CRF
Language	.910	.999	-
Compiler	.987	.998	.992
Optimization	.971	.993	.982
Version	.616	.910	.845

*

Toolchain details

[ISSTA 2011]

	Individually (SVM)	Linear CRF	Joint CRF
Language	.910	.999	-
Compiler	.987	.998	.992
Optimization	.971	.993	.982
Version	.616	.910	.845

*

slow, approximate inference

*

Toolchain details

[ISSTA 2011]

	Individually (SVM)	Linear CRF	Joint CRF
Language	.910	.999	-
Compiler	.987	.998	.992
Optimization	.971	.993	.982
Version	.616	.910	.845

*

*

slow, approximate inference

possibly inconsistent labels

C GCC 2005 LO
F77 MSVS 2008 HI

Para dyn™

Toolchain details

[ISSTA 2011]

	Individually (SVM)	Linear CRF	Joint CRF
Language	.910	.999	-
Compiler	.987	.998	.992
Optimization	.971	.993	.982
Version	.616	.910	.845

*

slow, approximate inference

possibly inconsistent labels

allows partial labels

*

C GCC 2005 LO
F77 MSVS 2008 HI

Toolchain details

[ISSTA 2011]

	Individually (SVM)	Linear CRF	Joint CRF
Language	.910	.999	-
Compiler	.987	.998	.992
Optimization	.971	.993	.982
Version	.616	.910	.845

MSVC

*

*

slow, approximate inference

possibly inconsistent labels

allows partial labels

C GCC 2005 LO
F77 MSVS 2008 HI

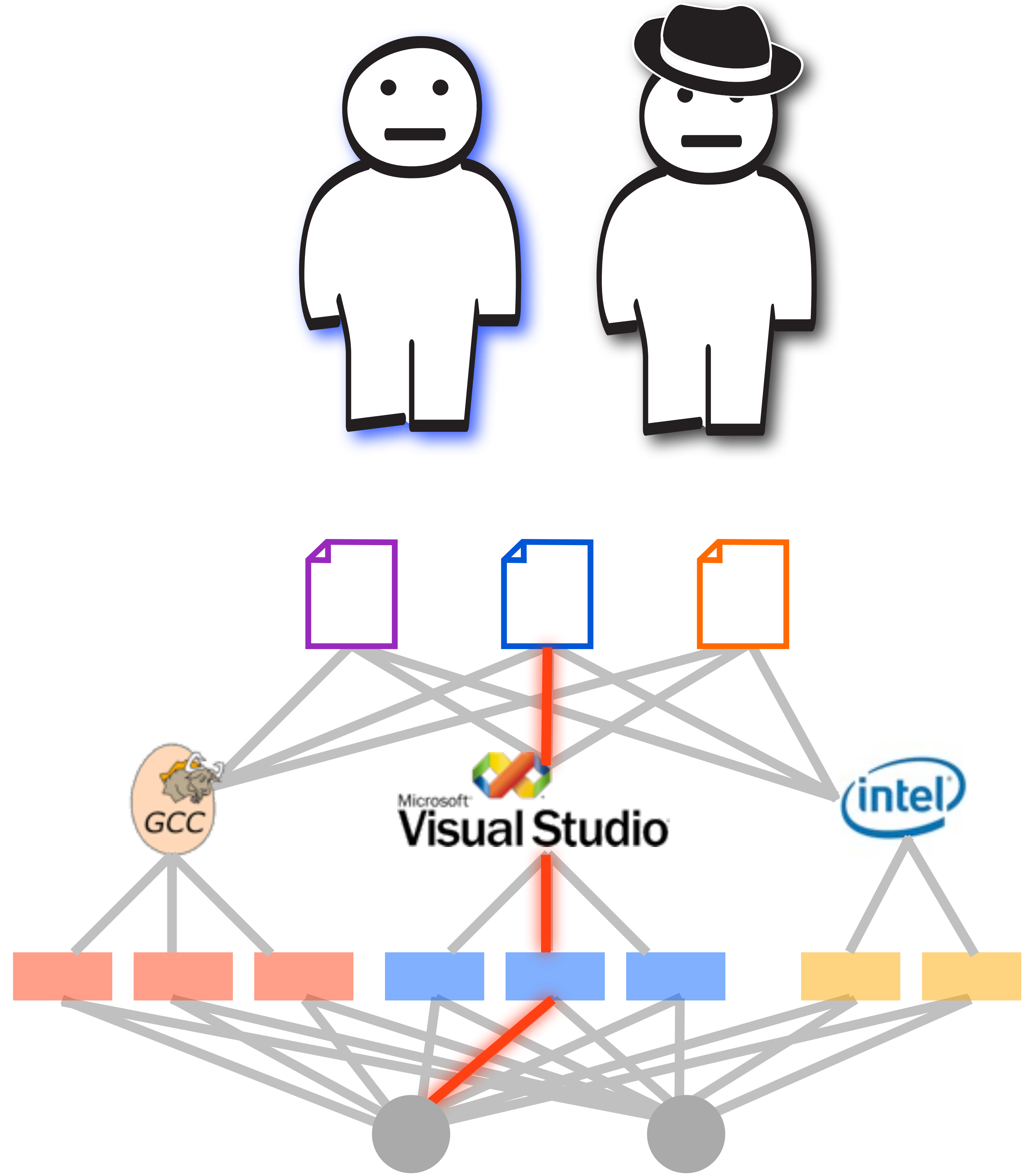
Outline

Program Modeling

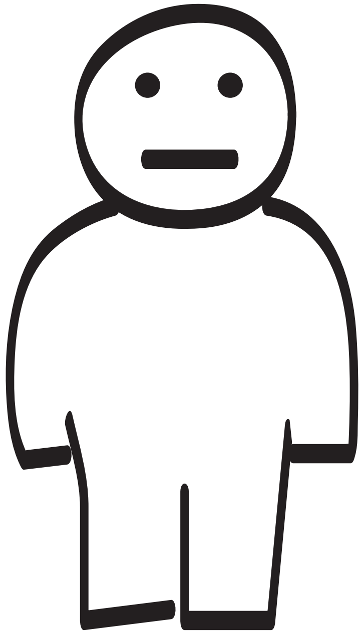
Compiler Toolchain

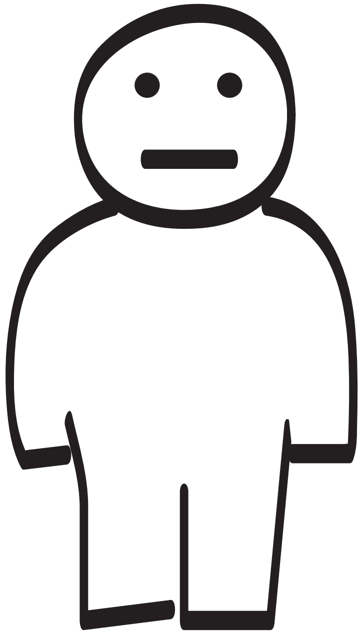
Authorship Attribution

Future Directions

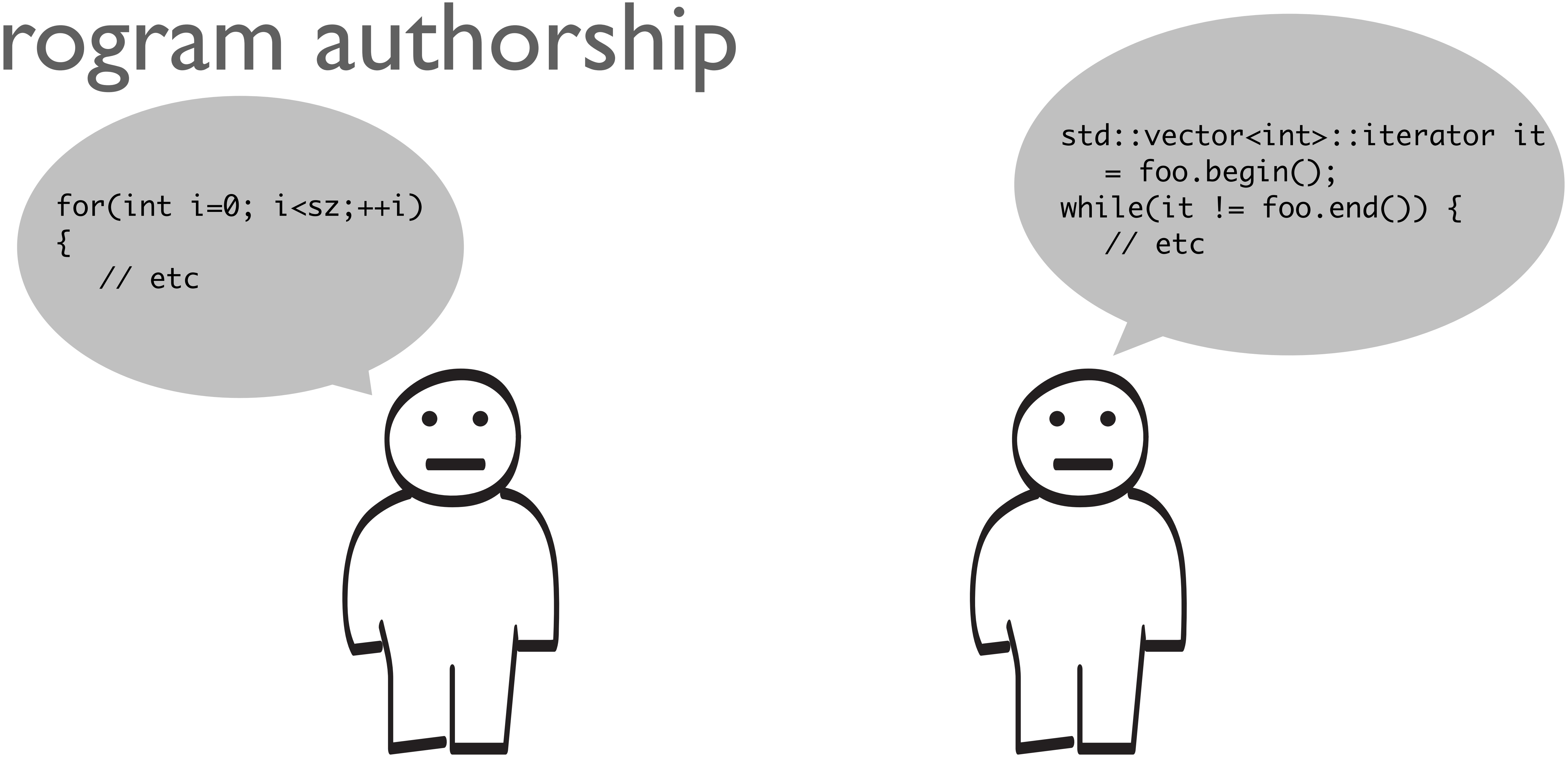


Program authorship





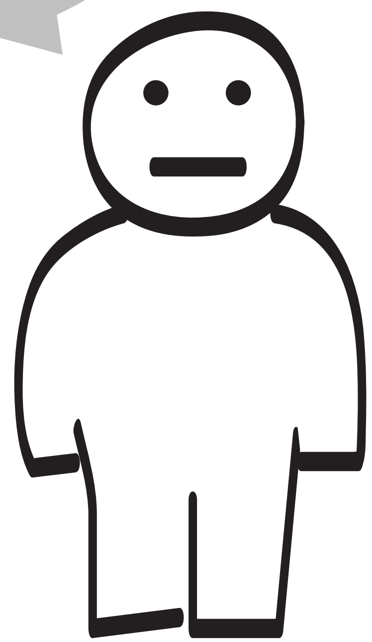
Program authorship

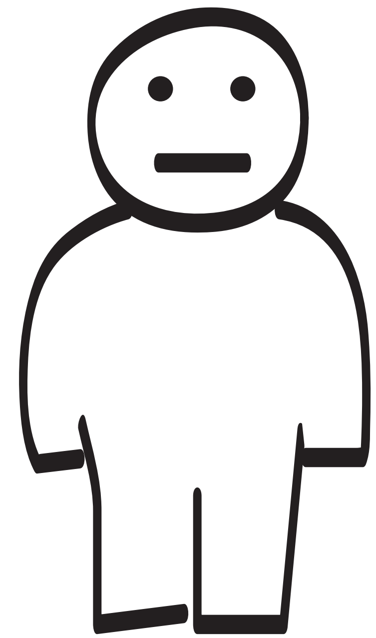


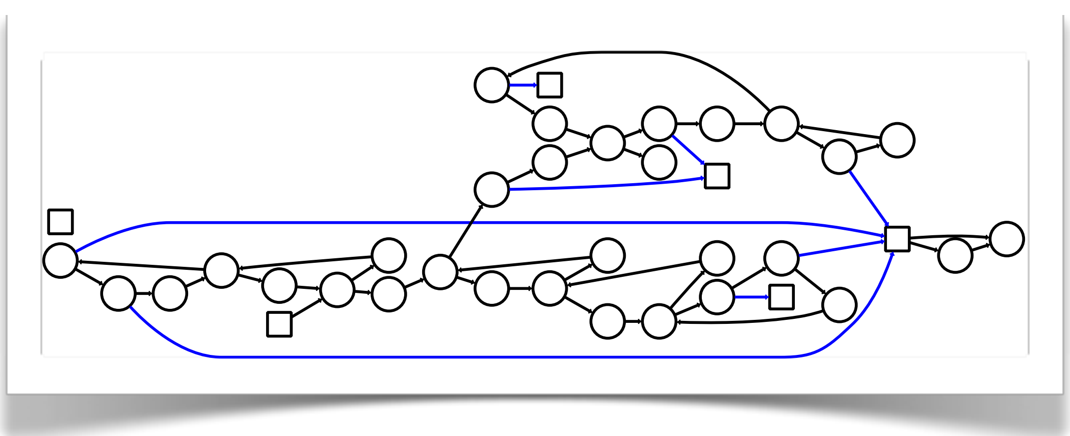
Program authorship

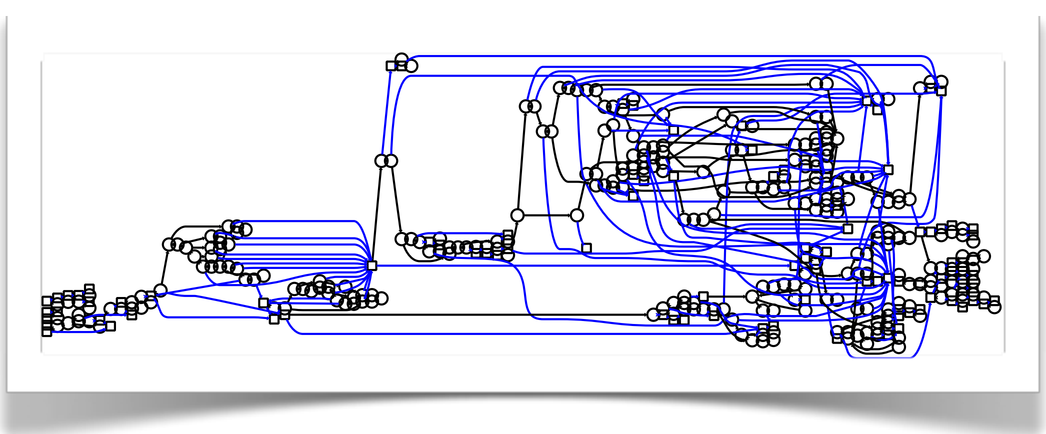
```
for(int i=0; i<sz;++i)
{
   // etc
```

```
std::vector<int>::iterator it
   = foo.begin();
while(it != foo.end()) {
   // etc
```

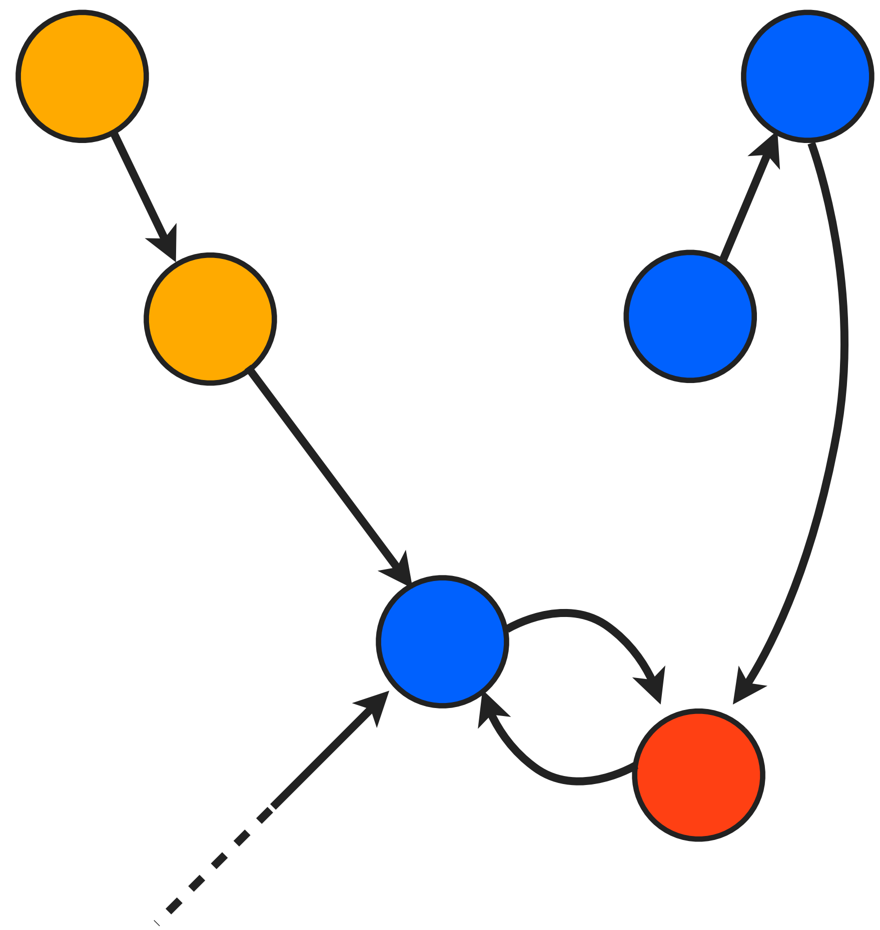




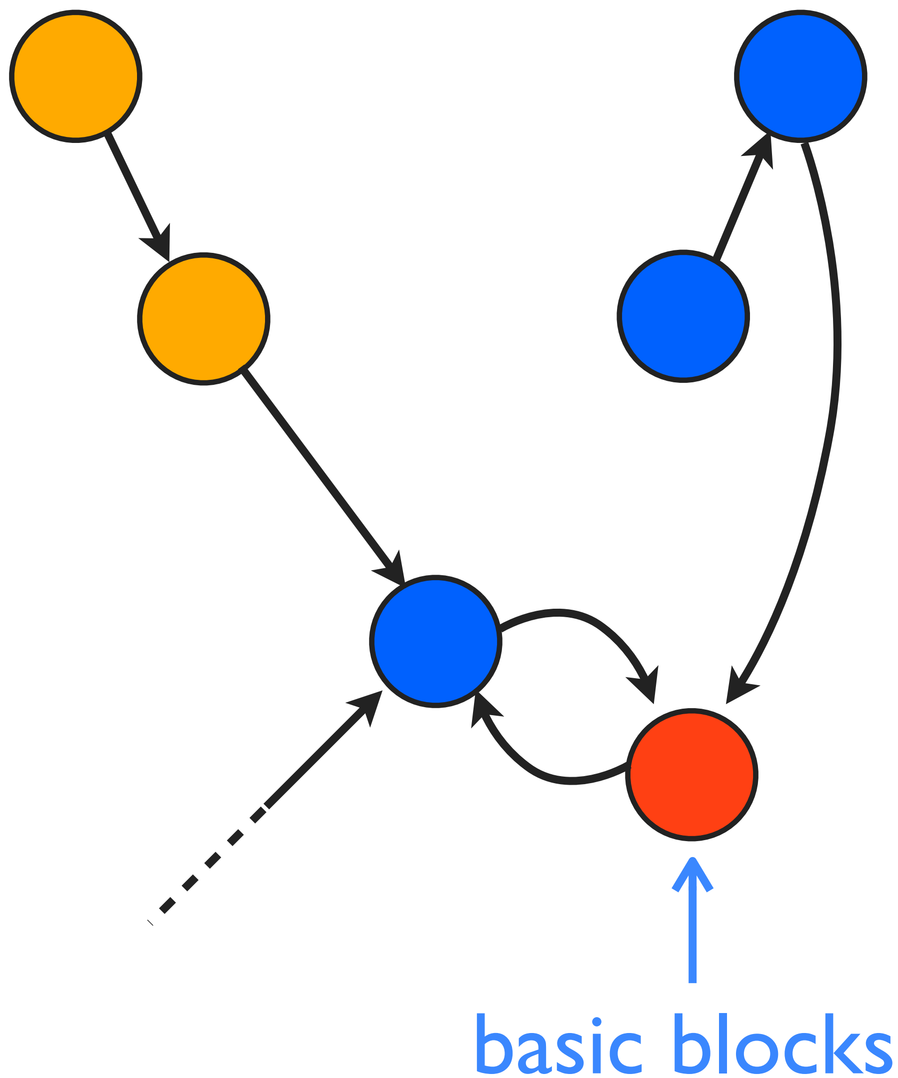




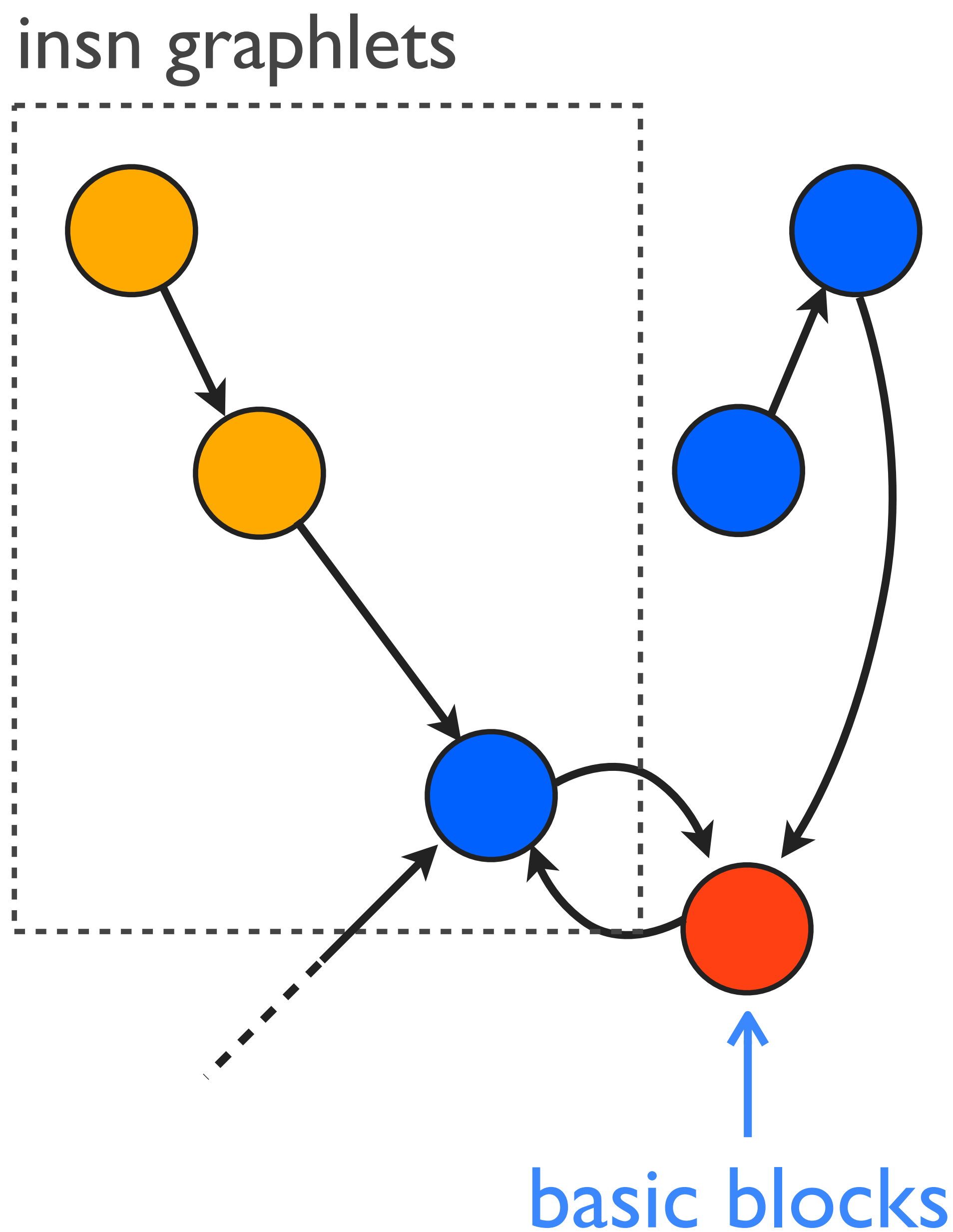
Long-range control flow



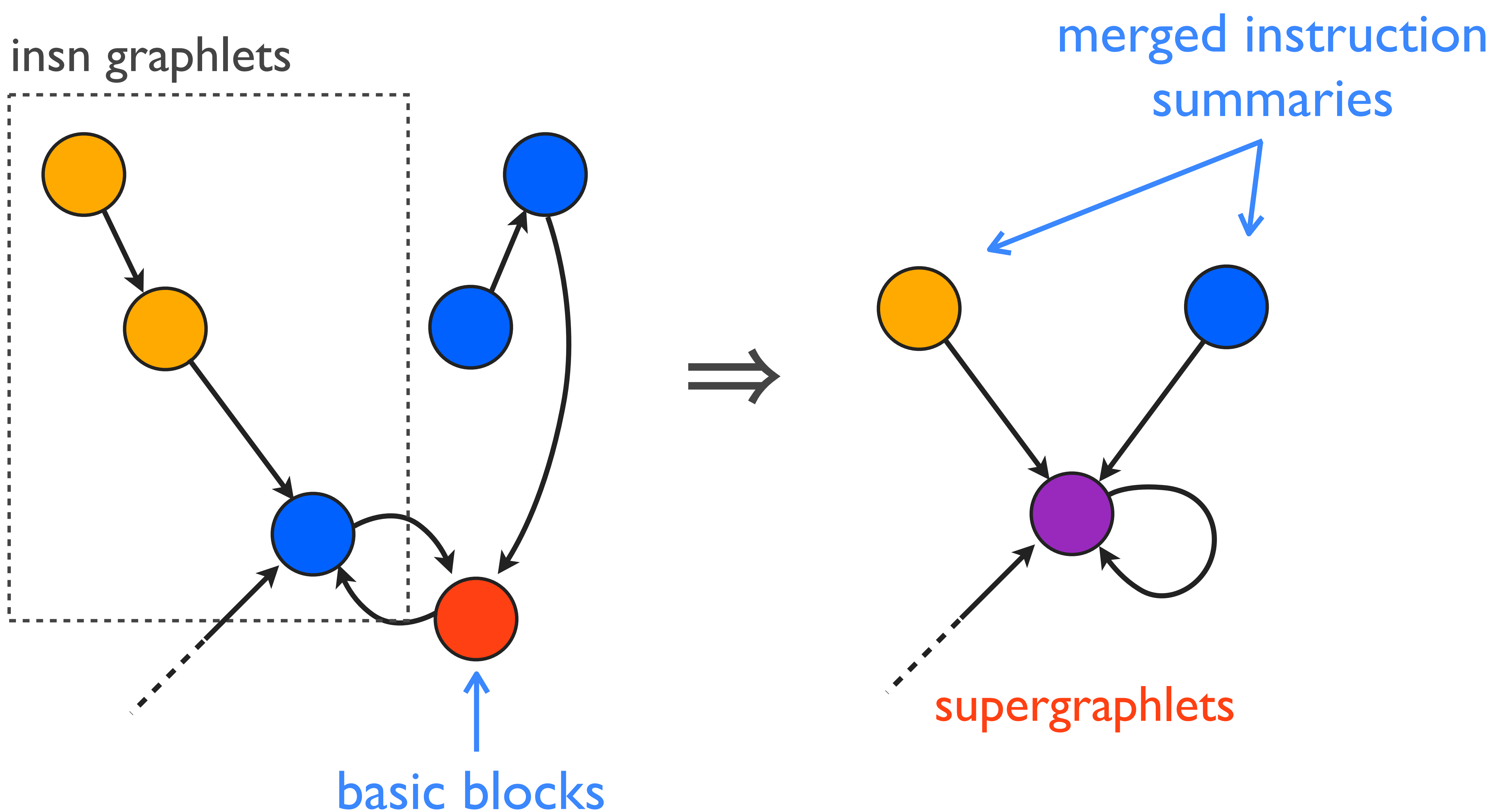
Long-range control flow



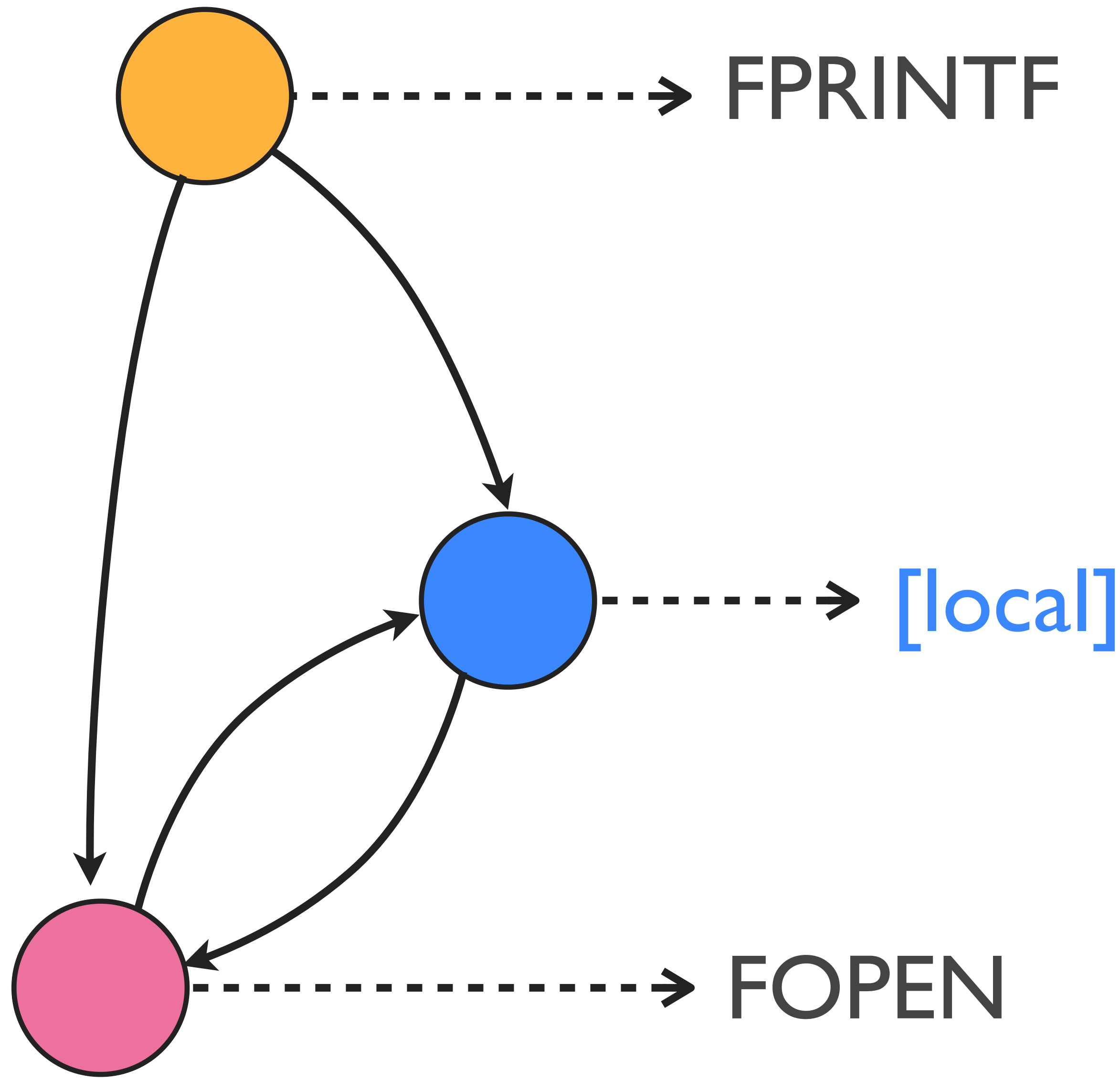
Long-range control flow



Long-range control flow



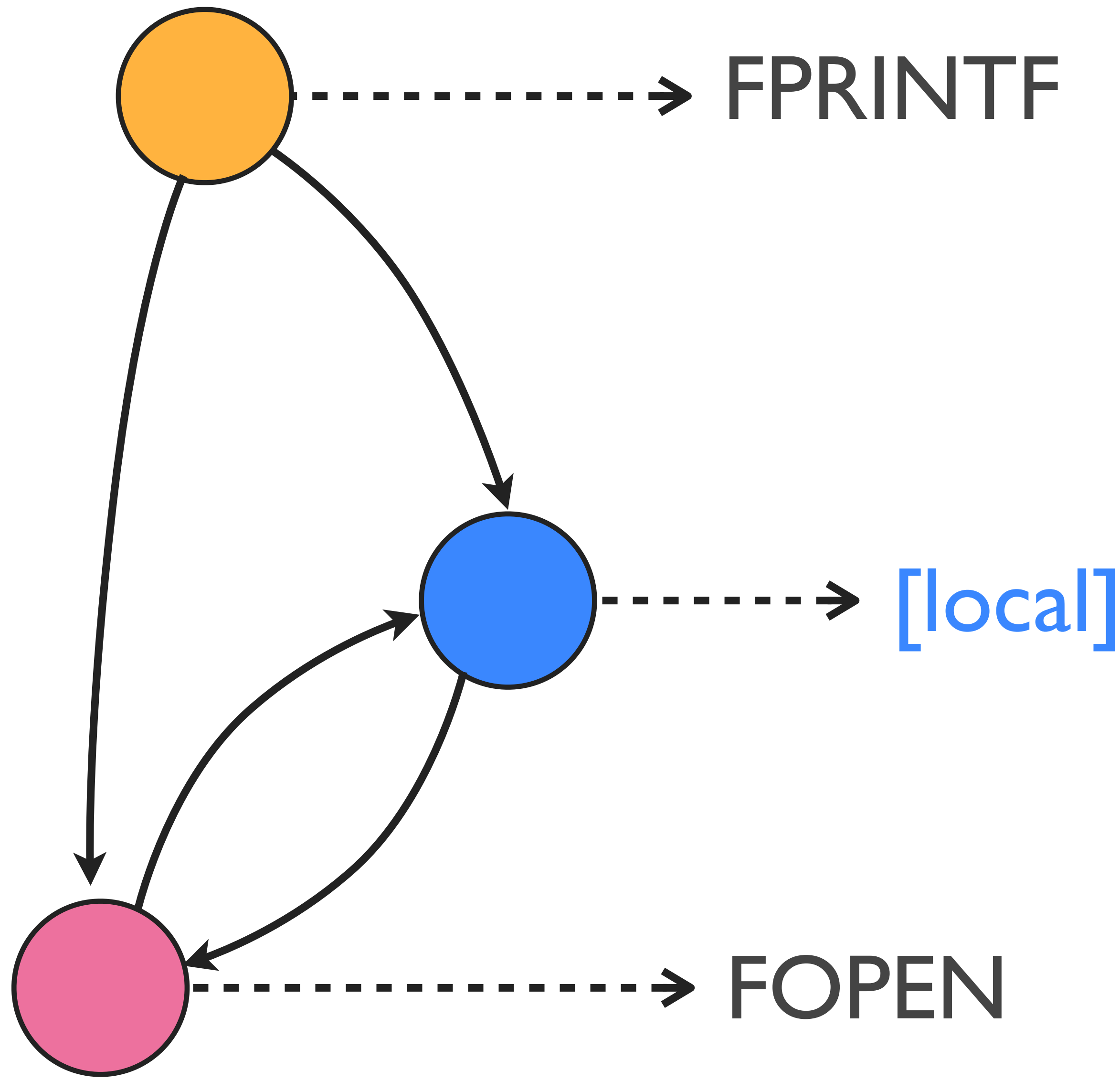
Interprocedural graphlets



Interprocedural graphlets

color ⇒ target

anonymous internal methods



Program-author dataset

Ideal:

1. Author labels	2. Parallel corpus	3. Linguistic homogeneity

Program-author dataset

Ideal:

1. Author labels
2. Parallel corpus
3. Linguistic homogeneity

code jam
hello, world!

WISCONSIN
UNIVERSITY OF WISCONSIN–MADISON
COMPUTER SCIENCES
(CS 537)

Program-author dataset

Ideal:

1. Author labels
2. Parallel corpus
3. Linguistic homogeneity

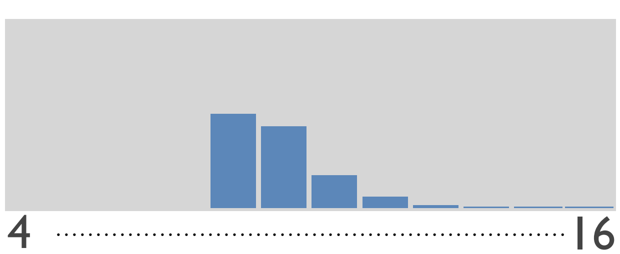
code jam

hello, world!

several contest years

C and C++ programs

8-16 programs per contestant



WISCONSIN

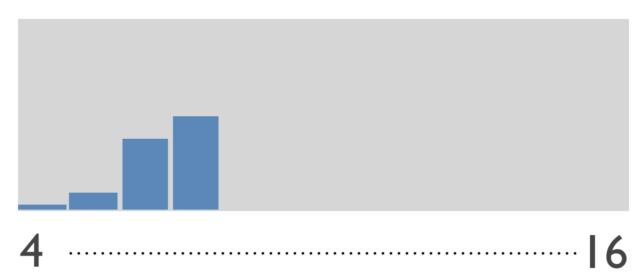
UNIVERSITY OF WISCONSIN–MADISON

COMPUTER SCIENCES

(CS 537)

C programs

some provided/template code



24

Author attribution

391,056 N-grams

54,705 idioms

152 library calls

37,358 graphlets

117,997 supergraphlets

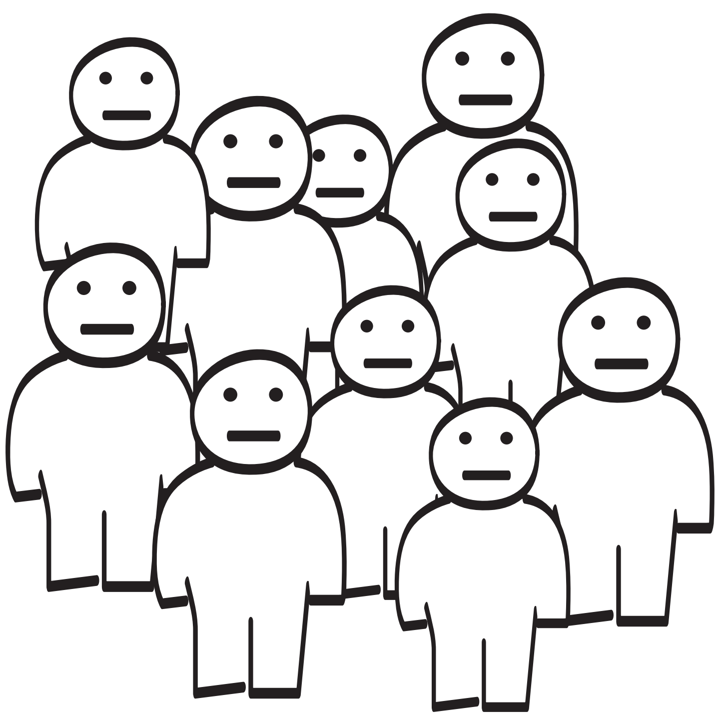
8,062 call graphlets

→ 1,900 features

Author attribution

391,056 N-grams
54,705 idioms
152 library calls
37,358 graphlets
117,997 supergraphlets
8,062 call graphlets

→ 1,900 features

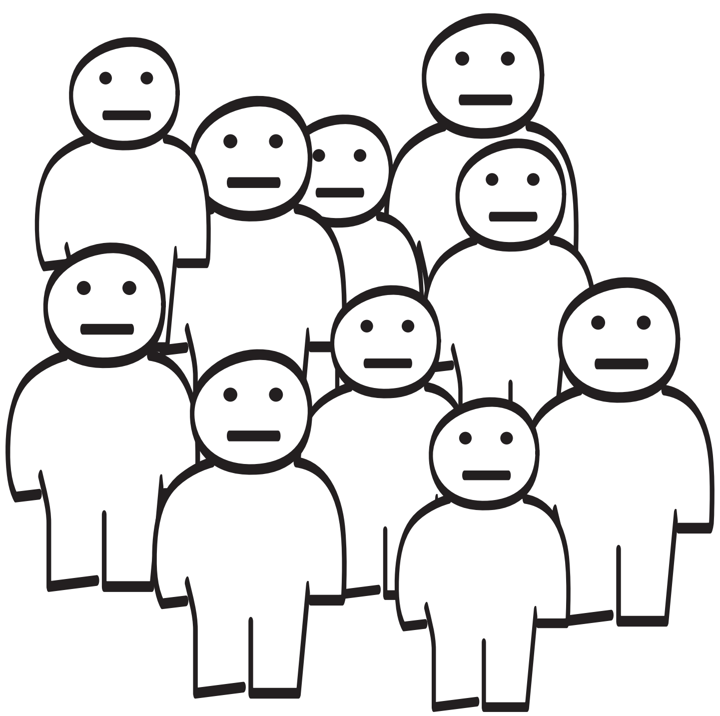


20 random programmers

Author attribution

391,056 N-grams
54,705 idioms
152 library calls
37,358 graphlets
117,997 supergraphlets
8,062 call graphlets

→ 1,900 features



20 random programmers
x 20 experiments

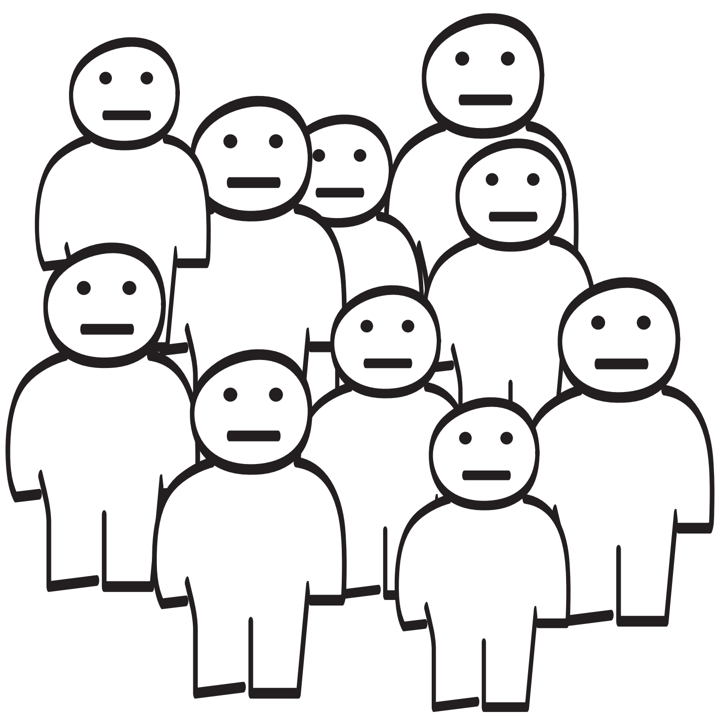
	CJ 2009	CJ 2010	CS 537
Exact	77.8%	76.8%	38.4%

25

Author attribution

391,056 N-grams
54,705 idioms
152 library calls
37,358 graphlets
117,997 supergraphlets
8,062 call graphlets

→ 1,900 features



20 random programmers

x 20 experiments

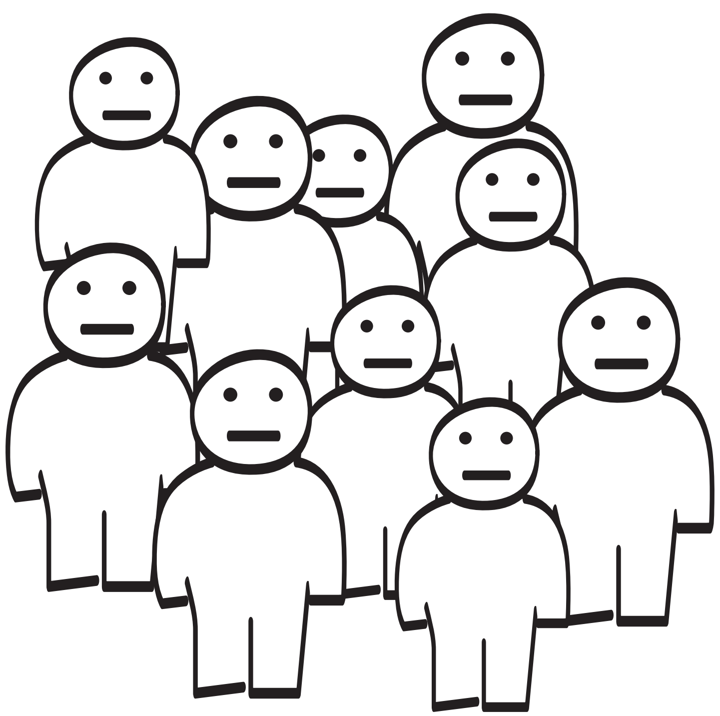
	CJ 2009	CJ 2010	CS 537
Exact	77.8%	76.8%	38.4%
Top-5	94.7%	93.7%	84.3%

Students have less distinctive styles?

Author attribution

391,056 N-grams
54,705 idioms
152 library calls
37,358 graphlets
117,997 supergraphlets
8,062 call graphlets

→ 1,900 features



20 random programmers
x 20 experiments

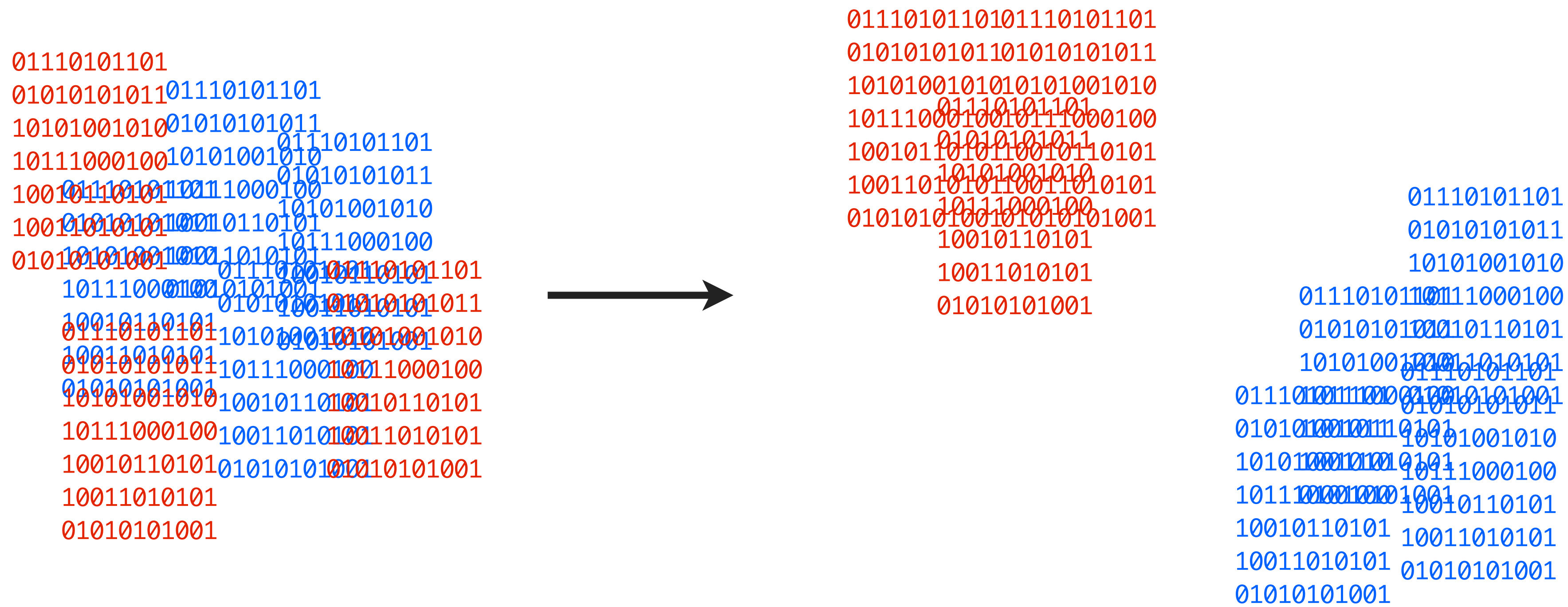
	CJ 2009	CJ 2010	CS 537
Exact	77.8%	76.8%	38.4%
Top-5	94.7%	93.7%	84.3%

Students have less distinctive styles?

1. CS537 has much less data
2. Template code + instructor guidance confound results

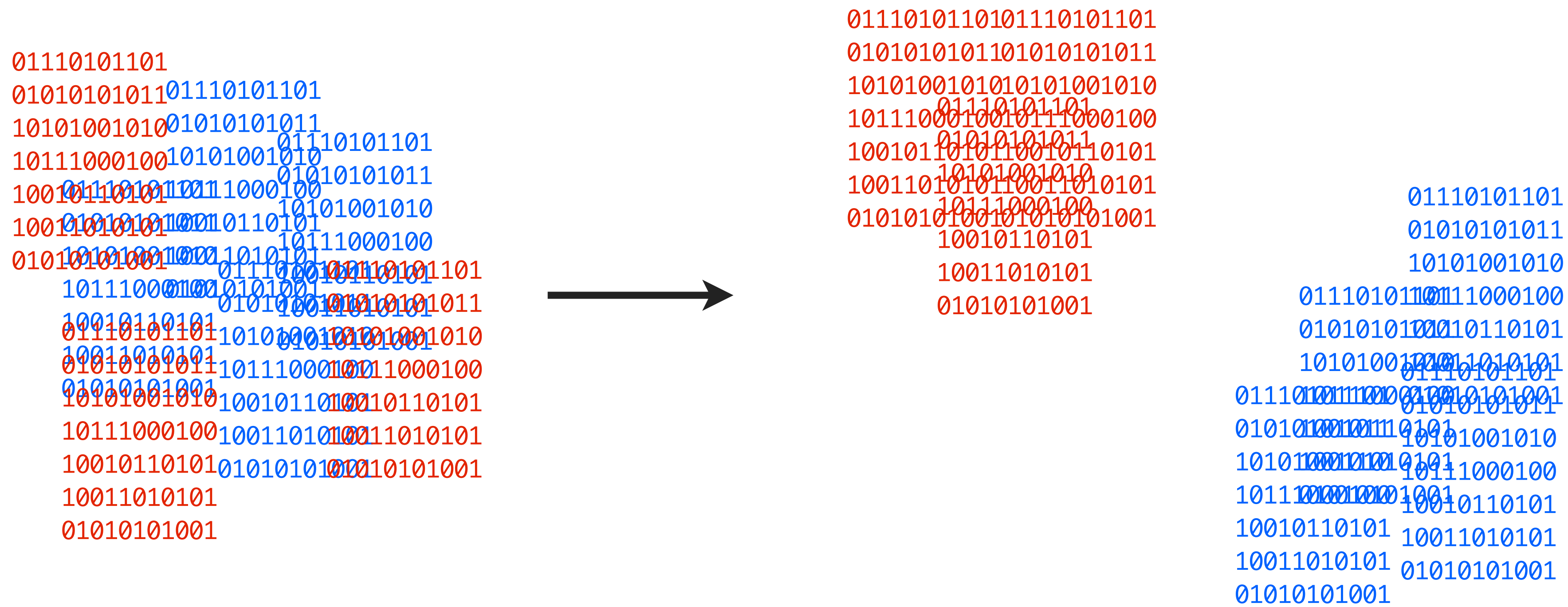
Style clustering

Programs, no training data

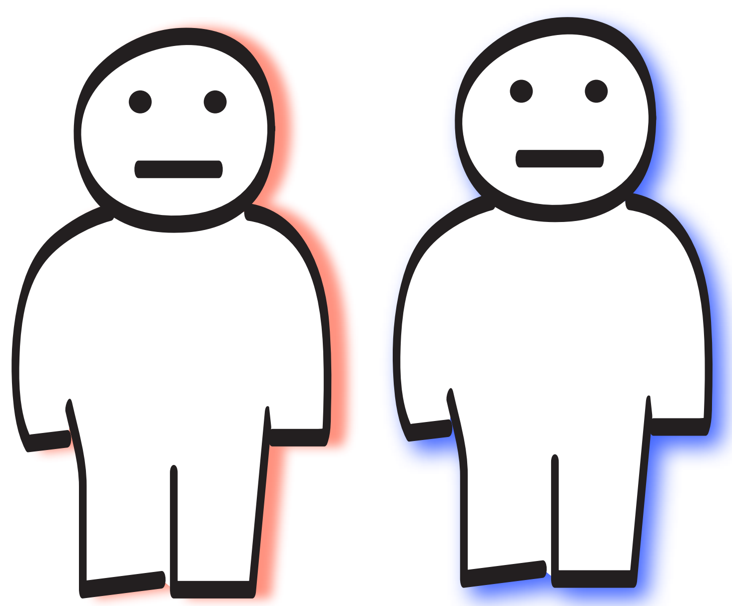


Style clustering

Programs, no training data

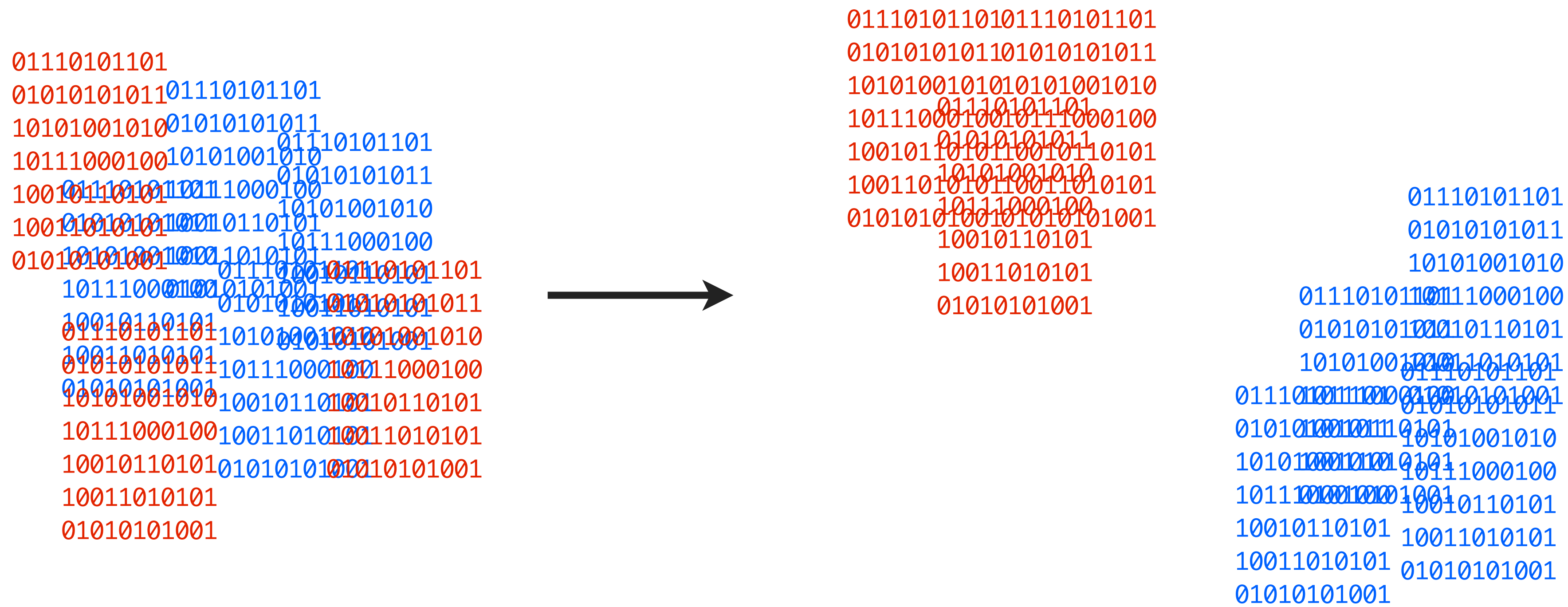


Conclude:

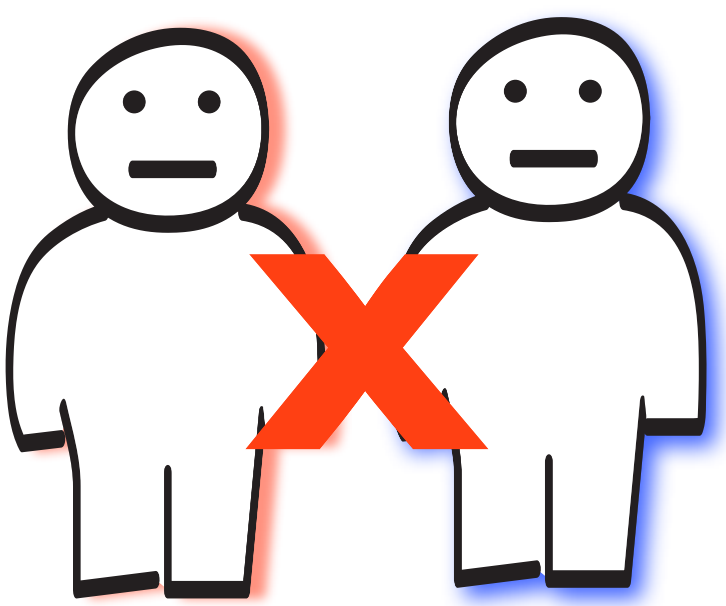


Style clustering

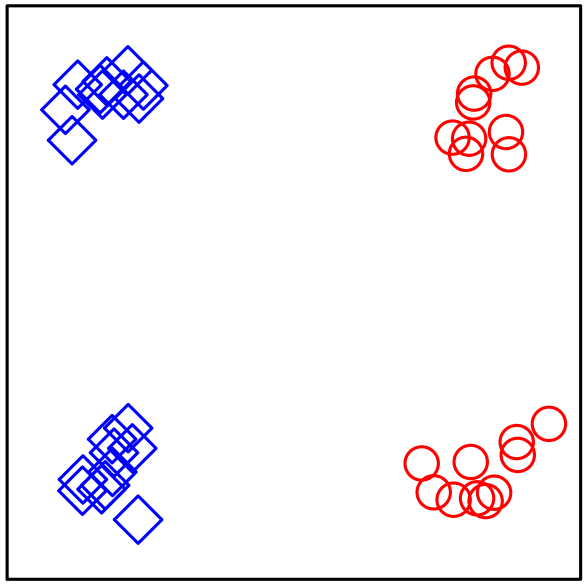
Programs, no training data



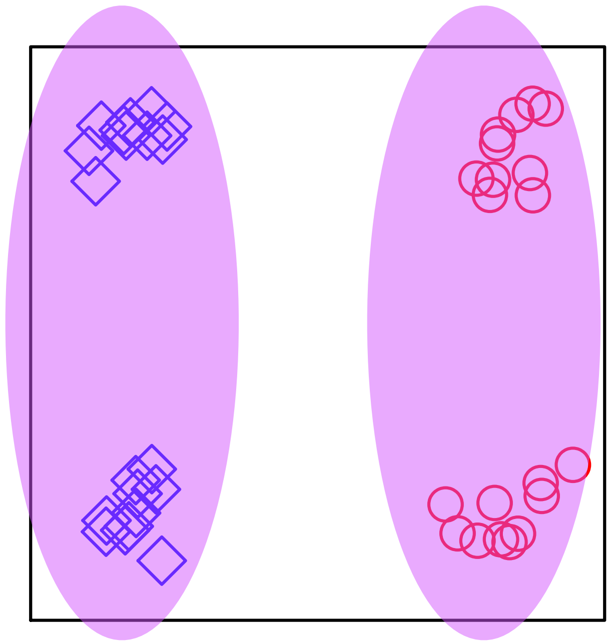
Conclude:



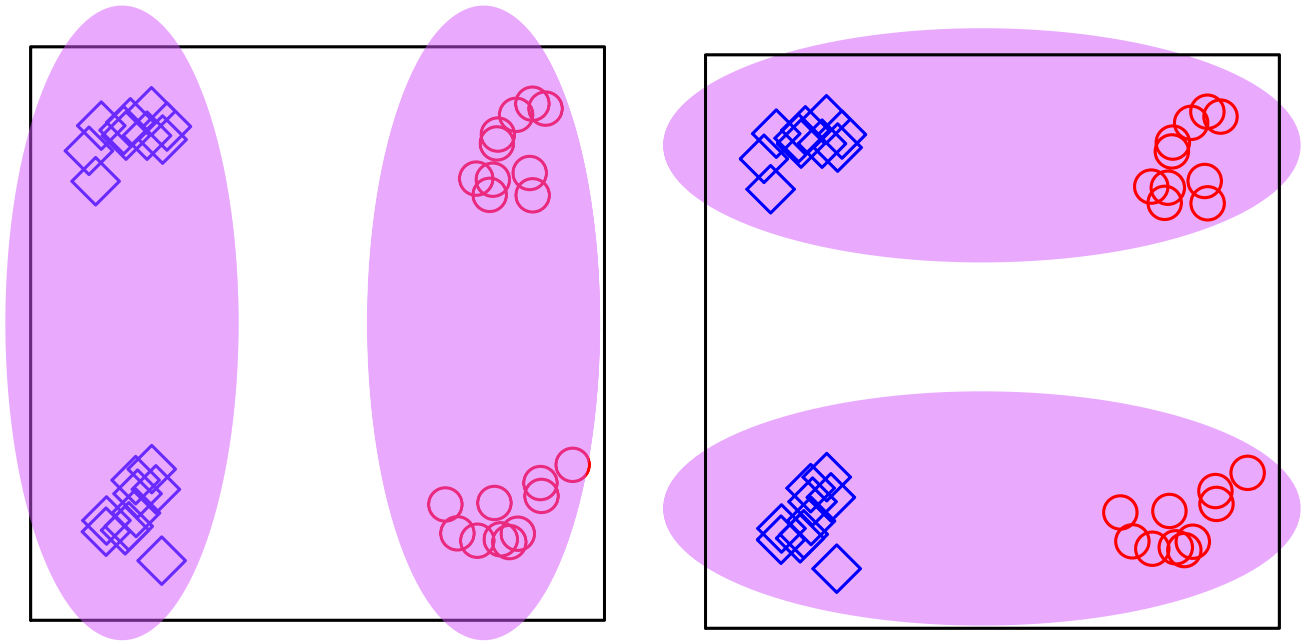
Distance metrics



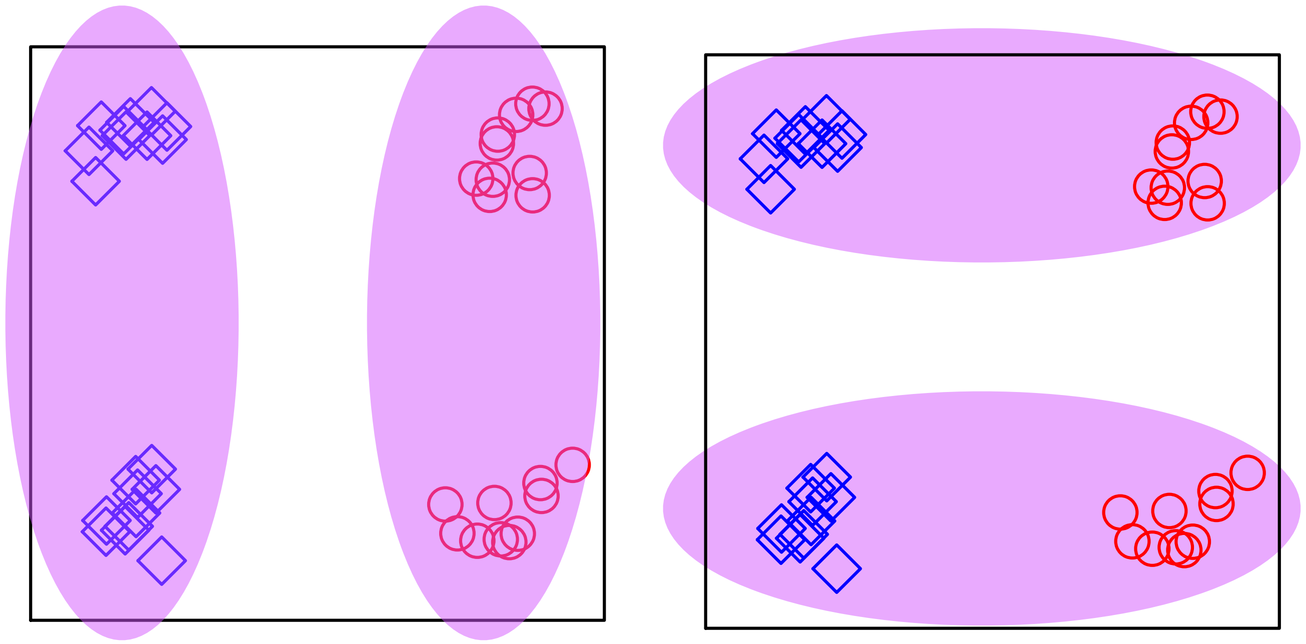
Distance metrics



Distance metrics



Distance metrics

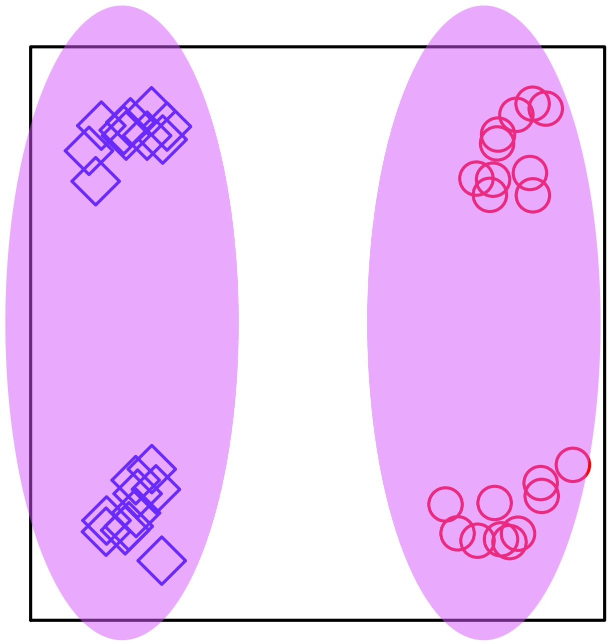


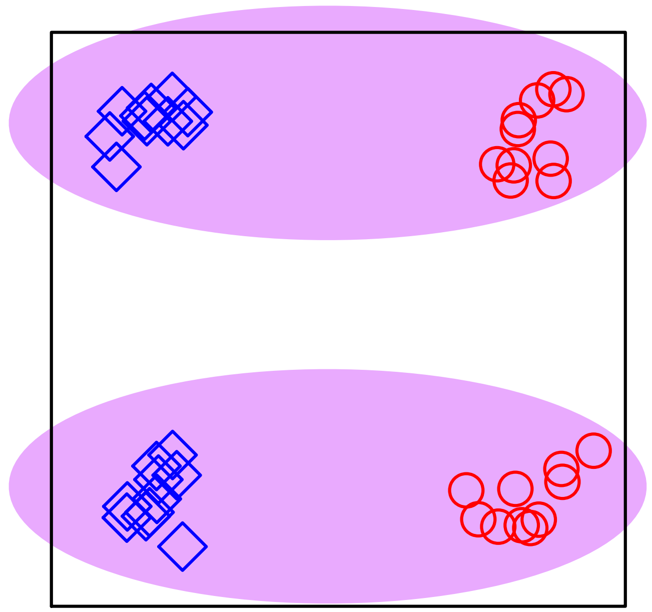
Mahalanobis distance

$$D_A(\mathbf{x}_a, \mathbf{x}_b) = \sqrt{(\mathbf{x}_a - \mathbf{x}_b)^T A (\mathbf{x}_a - \mathbf{x}_b)}$$

distance metric

Distance metrics





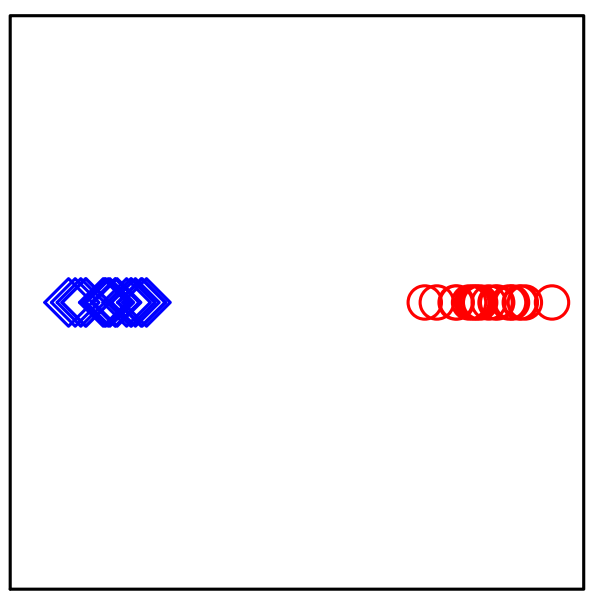
Mahalanobis distance

$$D_A(\mathbf{x}_a, \mathbf{x}_b) = \sqrt{(\mathbf{x}_a - \mathbf{x}_b)^T A (\mathbf{x}_a - \mathbf{x}_b)}$$

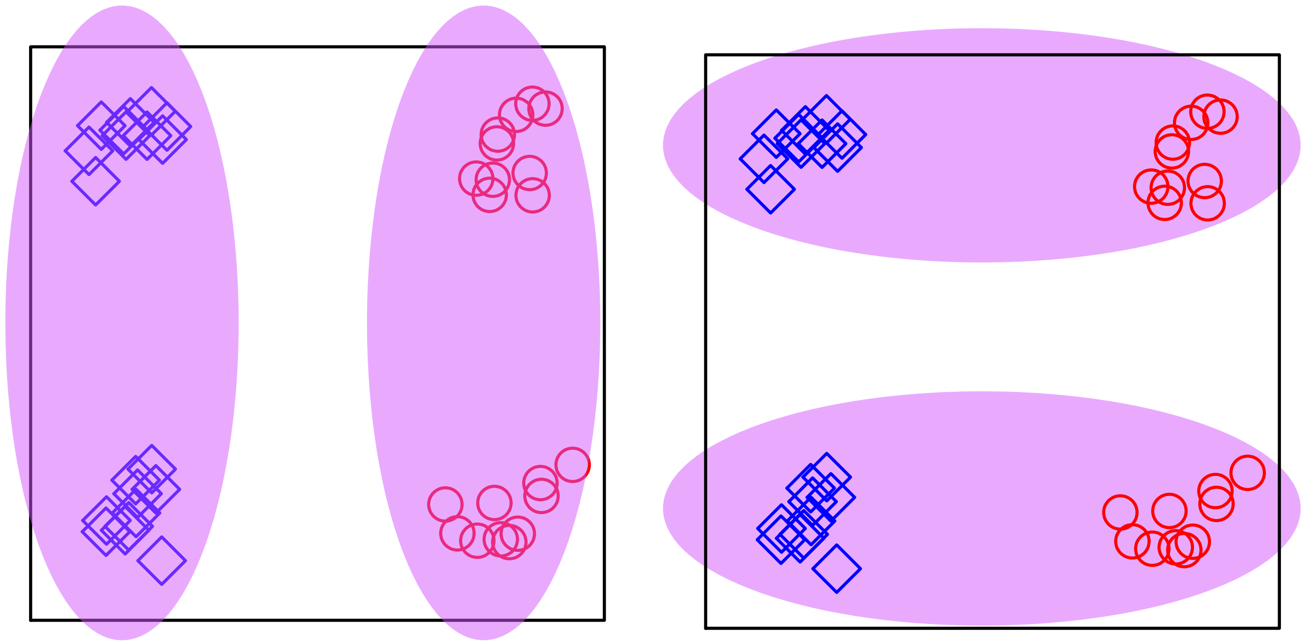
distance metric

Equivalently:

$$\mathbf{x}^T \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$$



Distance metrics



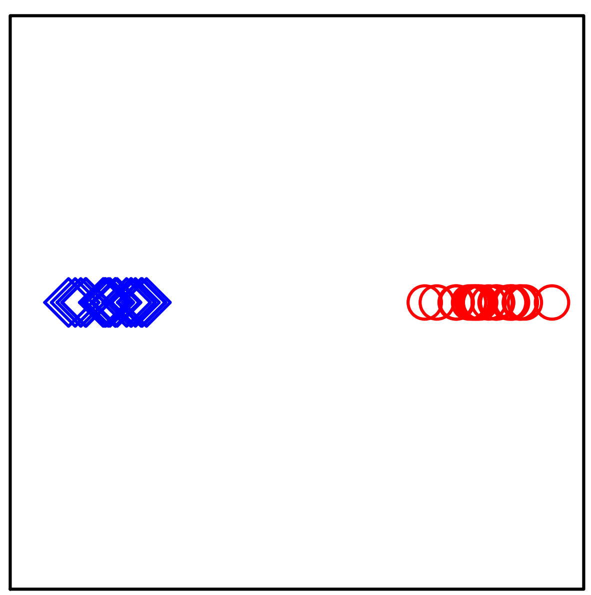
Mahalanobis distance

$$D_A(\mathbf{x}_a, \mathbf{x}_b) = \sqrt{(\mathbf{x}_a - \mathbf{x}_b)^T A (\mathbf{x}_a - \mathbf{x}_b)}$$

distance metric

Equivalently:

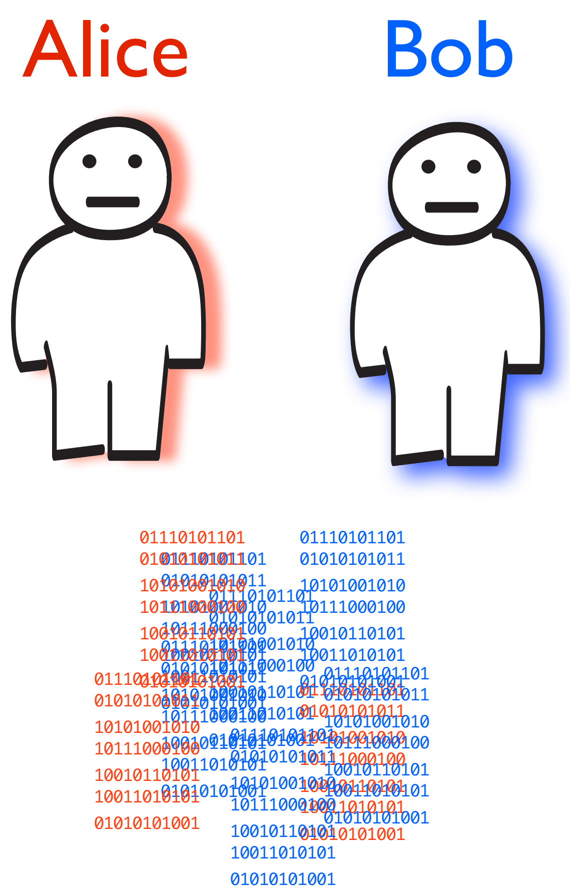
$$\mathbf{x}^T \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$$



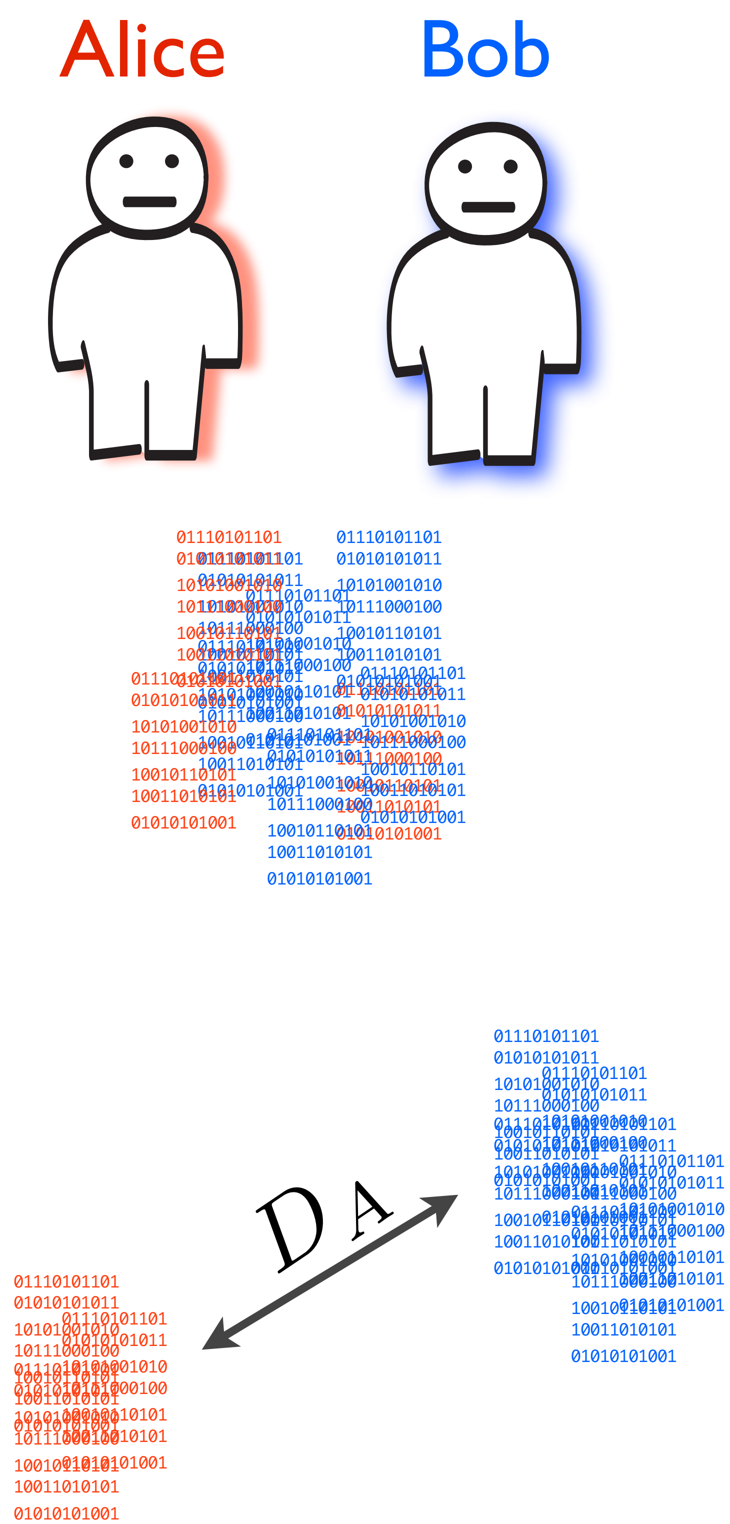
How do we get A?

27

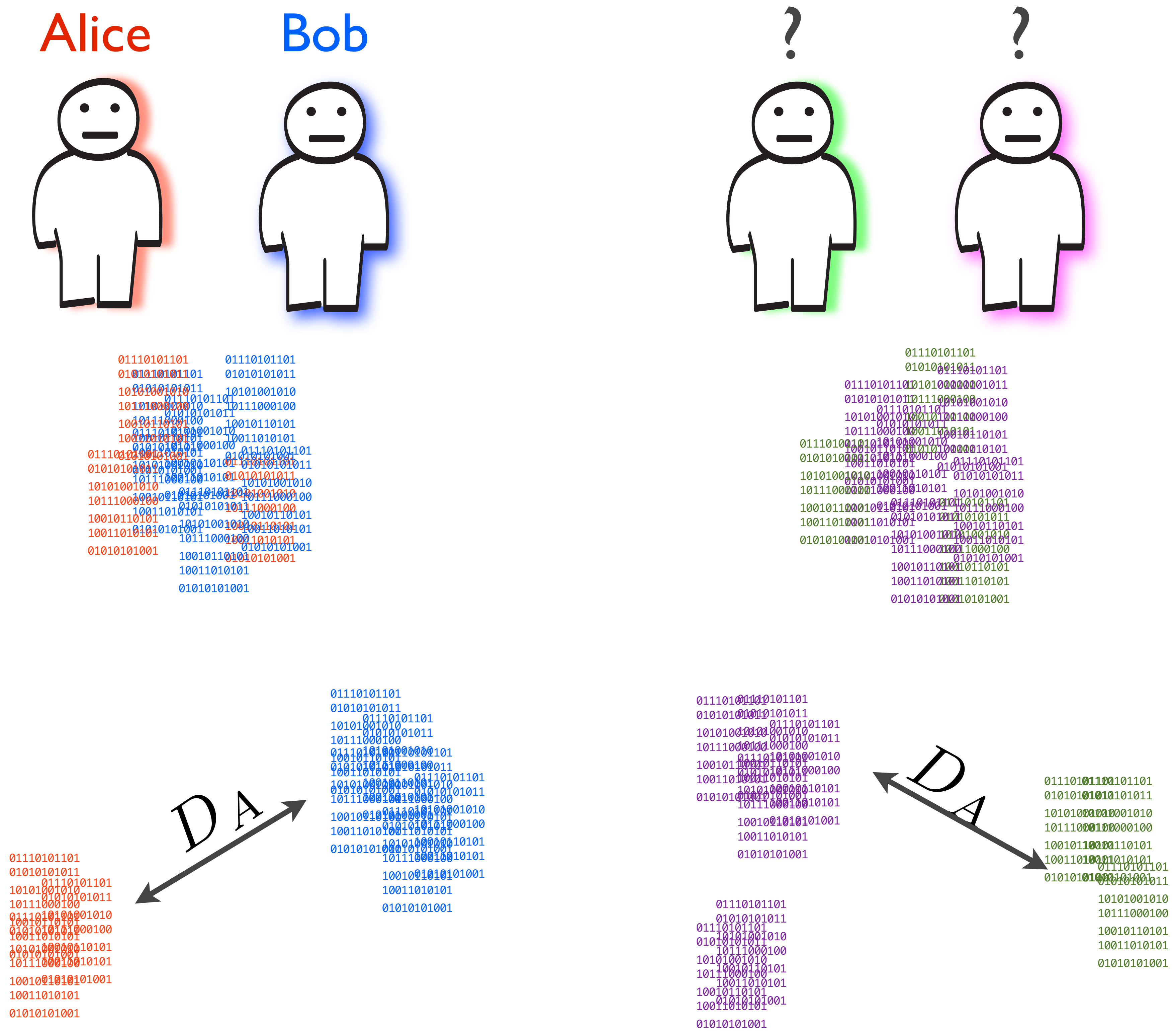
Transfer learning



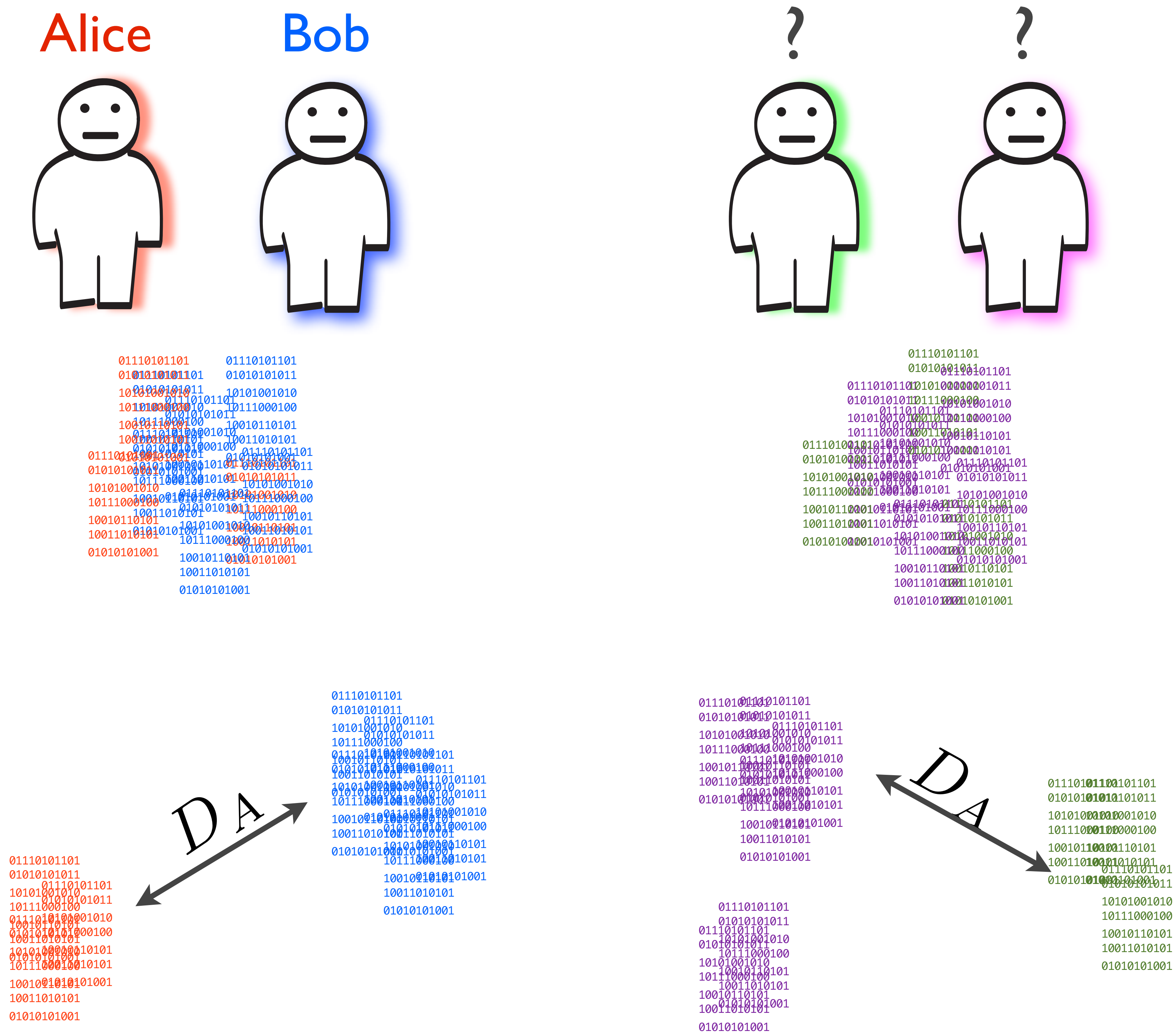
Transfer learning



Transfer learning



Transfer learning



Large-margin Nearest Neighbors (LMNN)

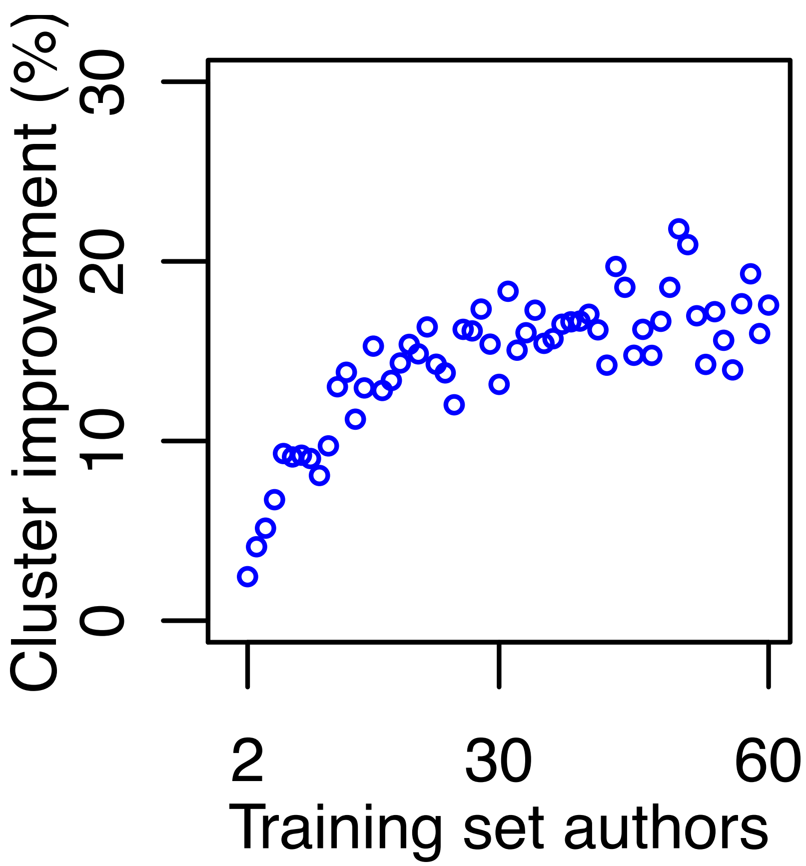
Weinberger, Saul 2009

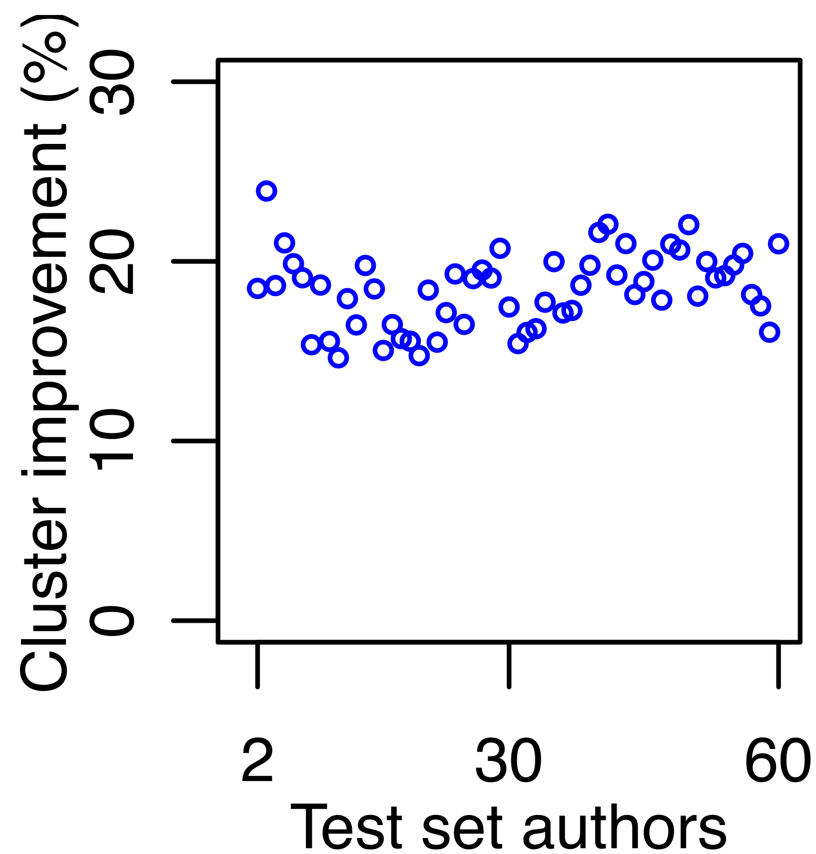
semi-definite program ☹

one-time cost ☺

Clustering (Code Jam 2010)

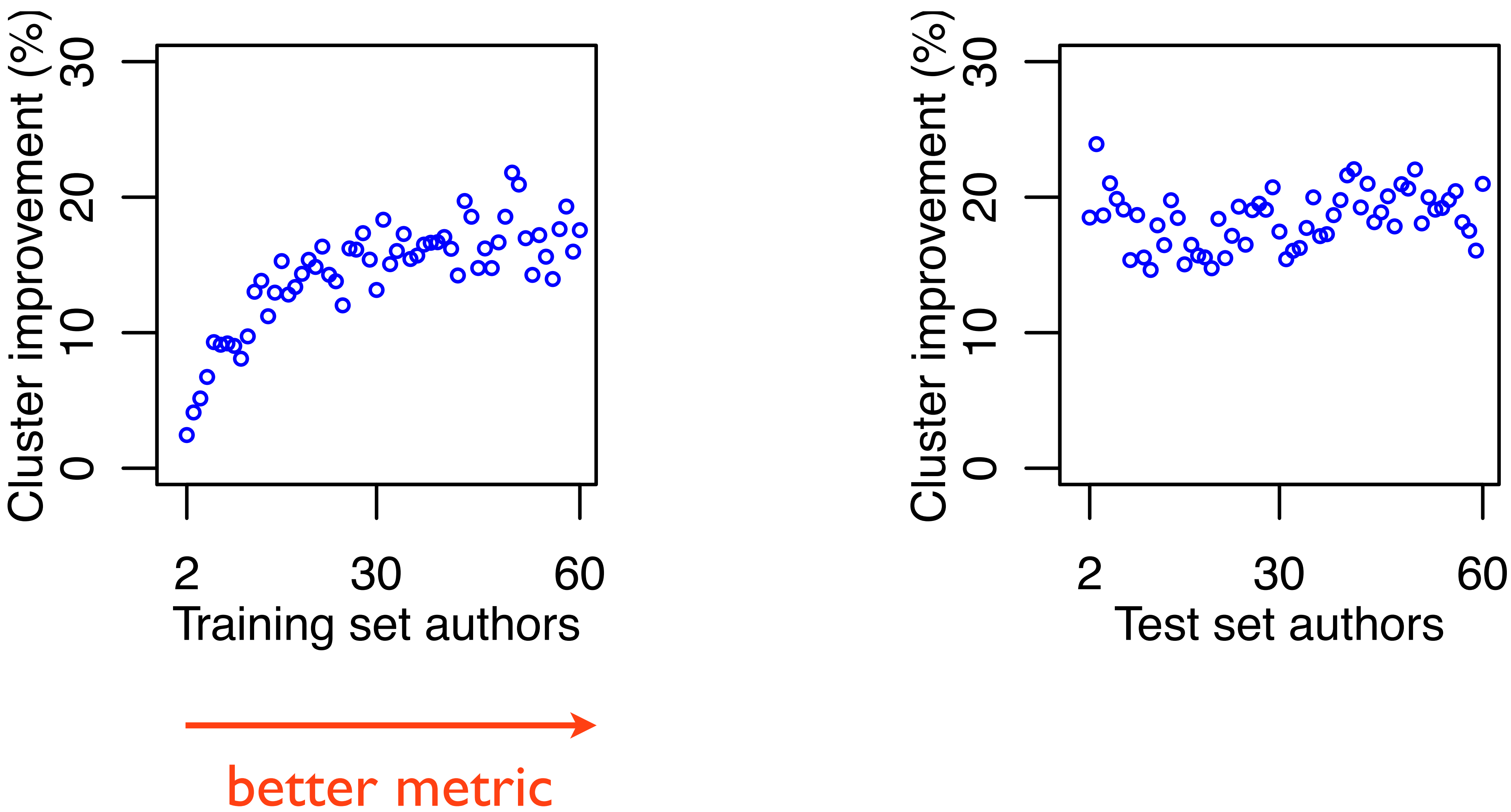
Compare Euclidean vs. Mahalanobis distance





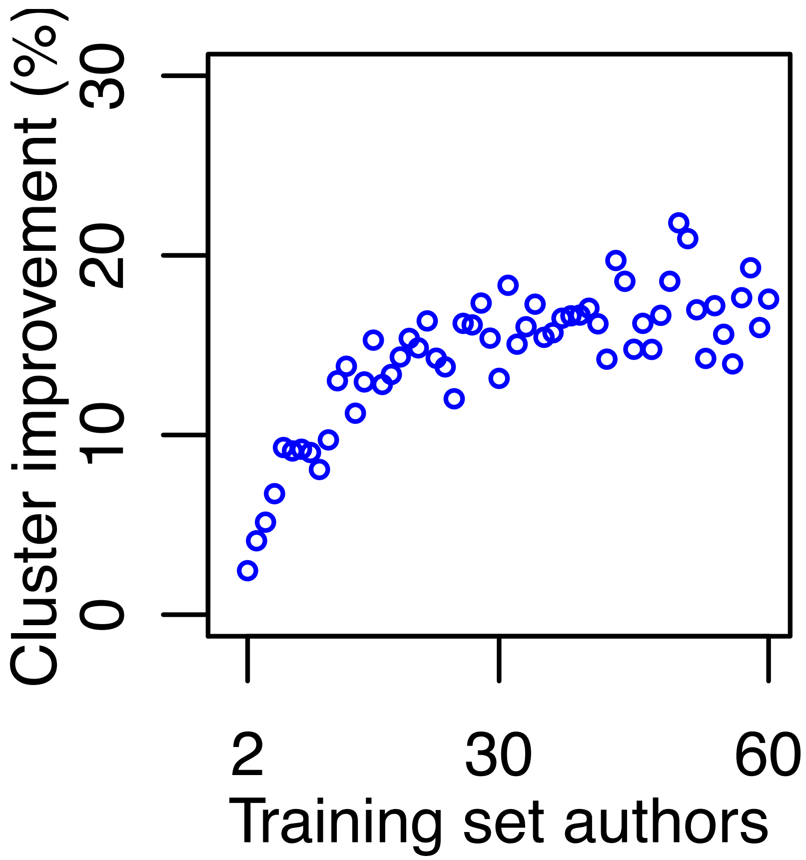
Clustering (Code Jam 2010)

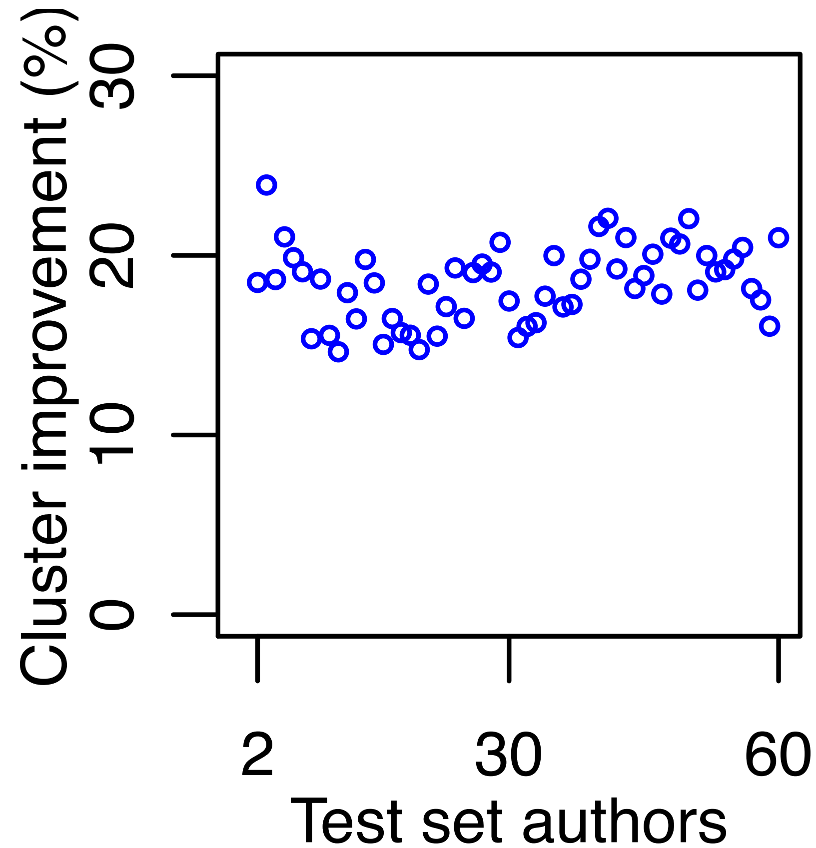
Compare Euclidean vs. Mahalanobis distance



Clustering (Code Jam 2010)

Compare Euclidean vs. Mahalanobis distance



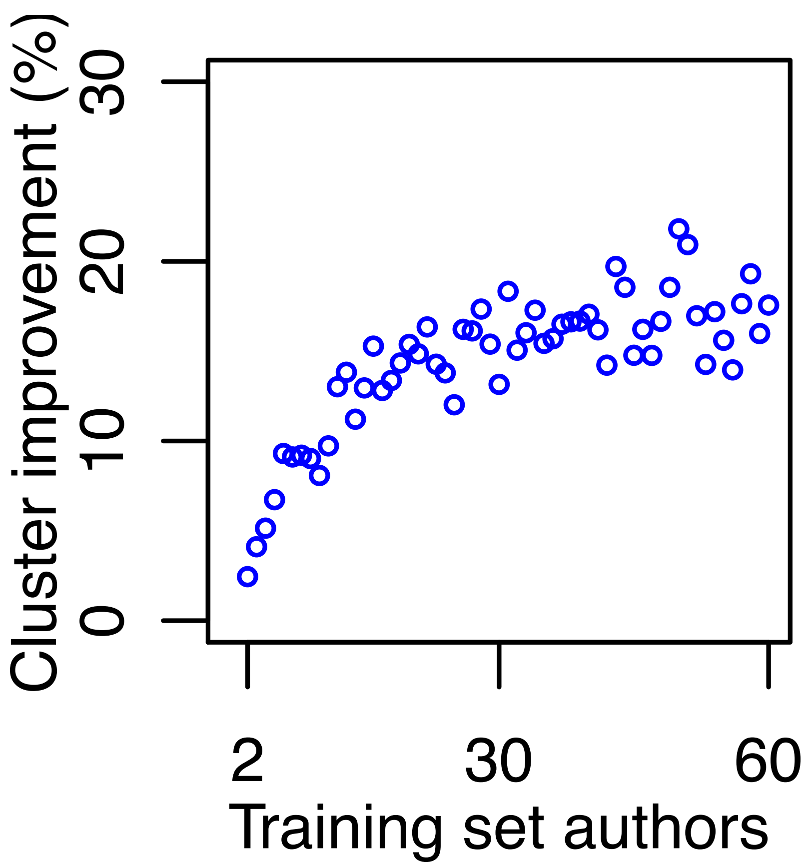


better metric

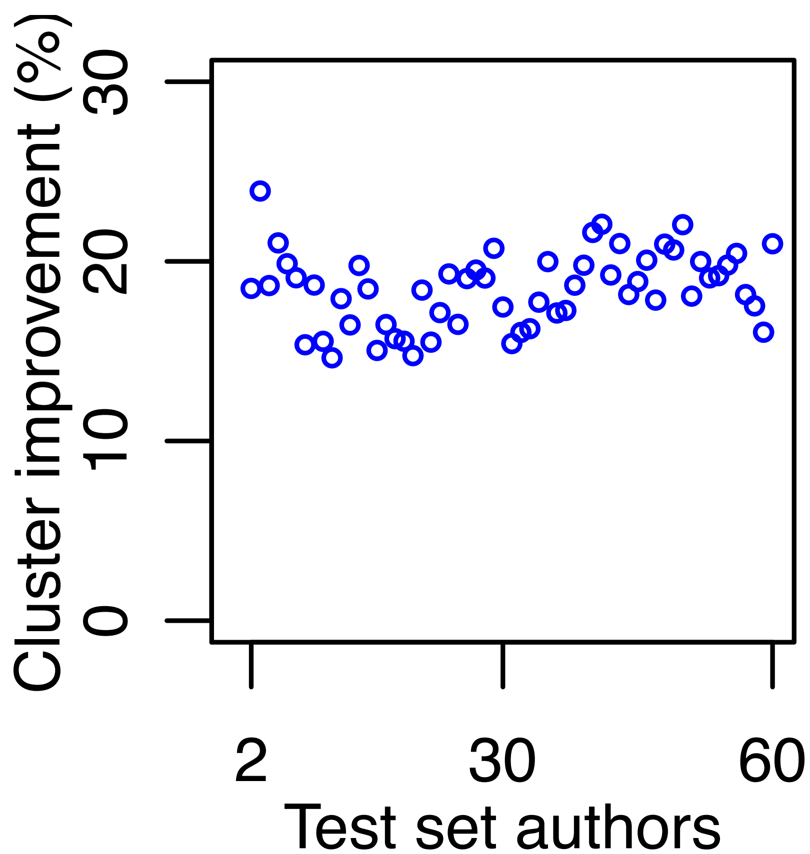
stable relative improvement

Clustering (Code Jam 2010)

Compare Euclidean vs. Mahalanobis distance



better metric



stable relative improvement

Average .723 NMI clustering 10 authors' programs

[0,1] → 1 is better

Outline

Program Modeling

Compiler Toolchain

Authorship Attribution

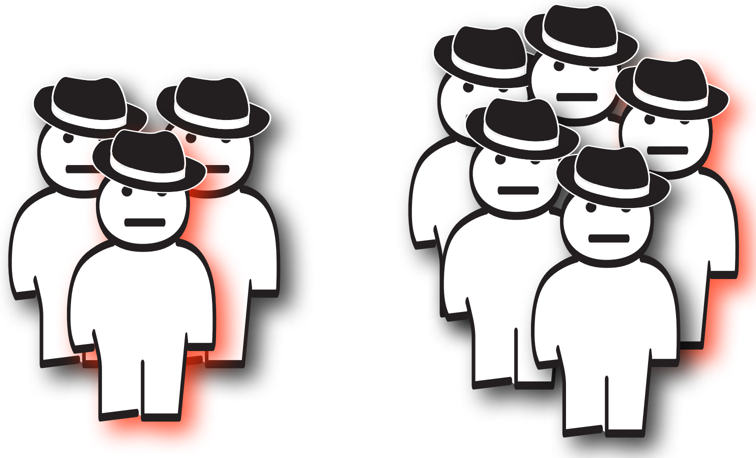
Future Directions

Component models

semi-open world provenance

component sharing (e.g. command and control)

programmer movement between groups

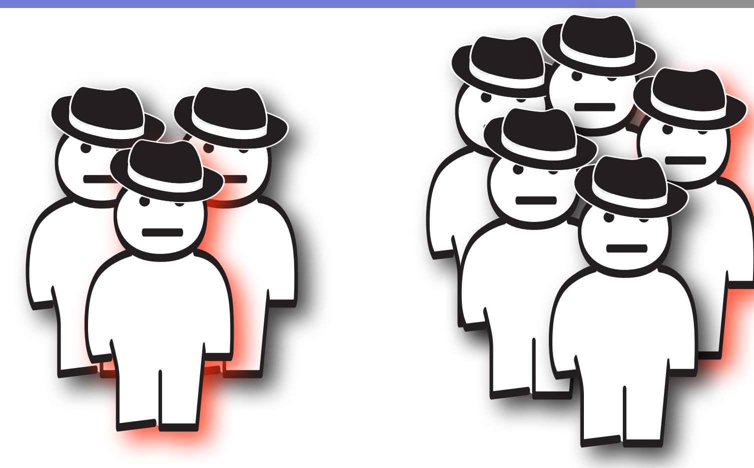


Component models

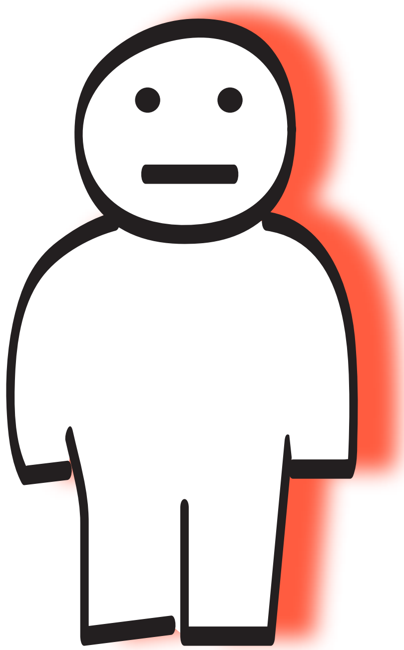
semi-open world provenance

component sharing (e.g. command and control)

programmer movement between groups



mixture of styles

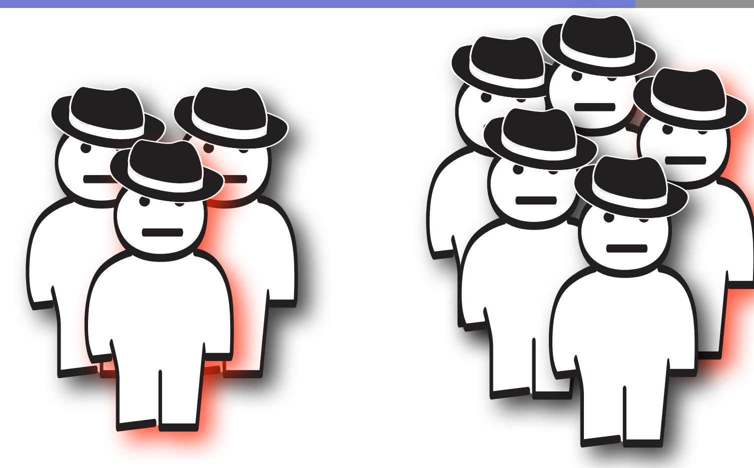


Component models

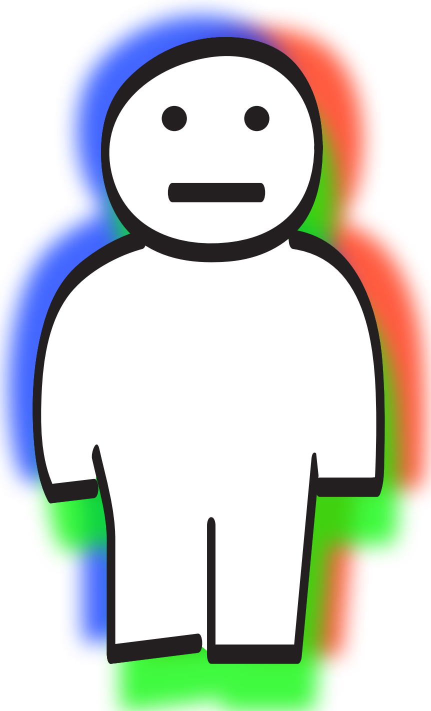
semi-open world provenance

component sharing (e.g. command and control)

programmer movement between groups



mixture of styles



01110101101	01110101101
01010101011	01010101011
10101001010	10101001010
10111000100	10111000100
10010110101	10010110101
10011010101	10011010101
01010101001	01010101001

style vs. functionality?

infinite mixture models

interpreting style clusters

Para dyn™

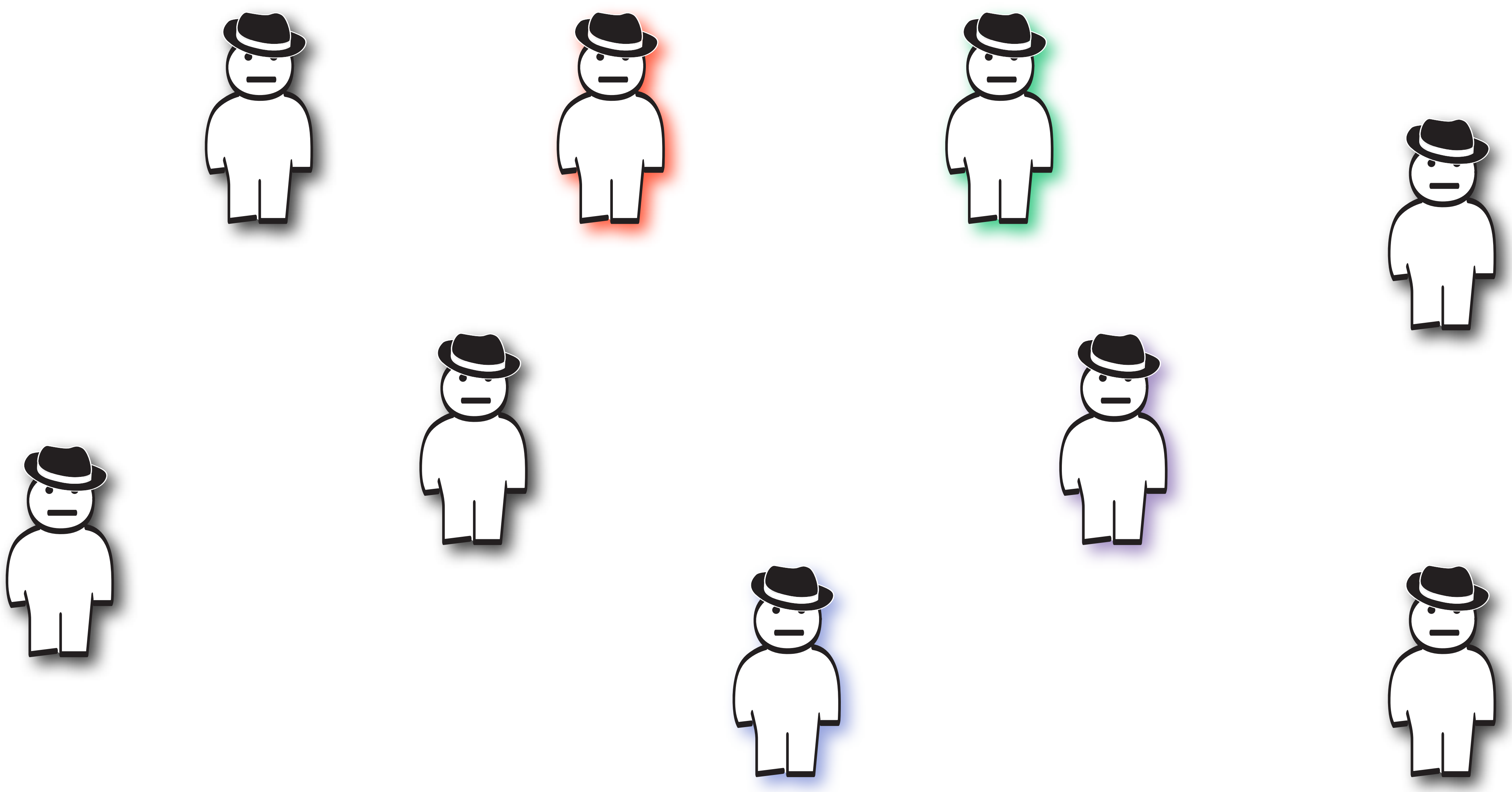
Social code networks

program binaries

32

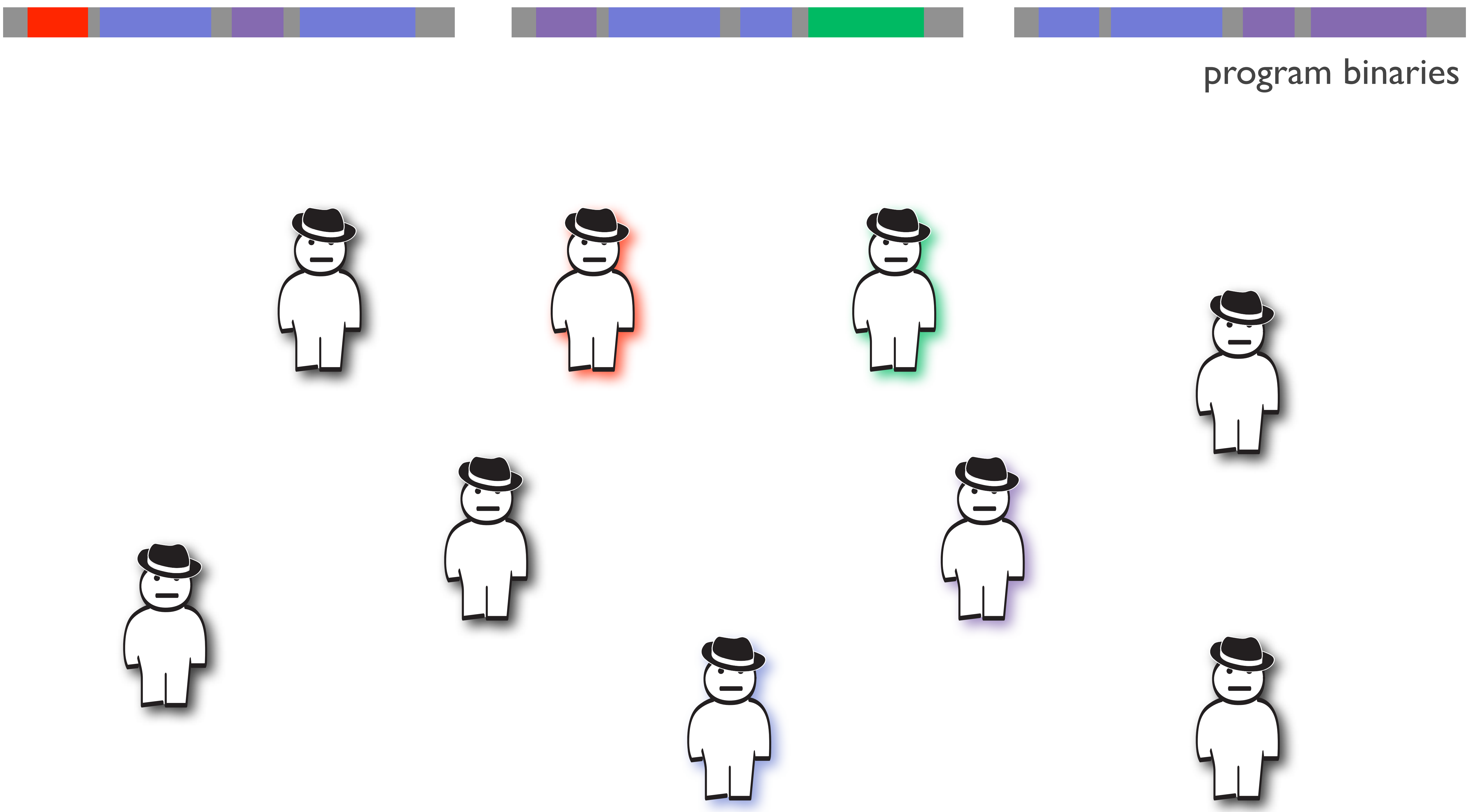
Social code networks

program binaries



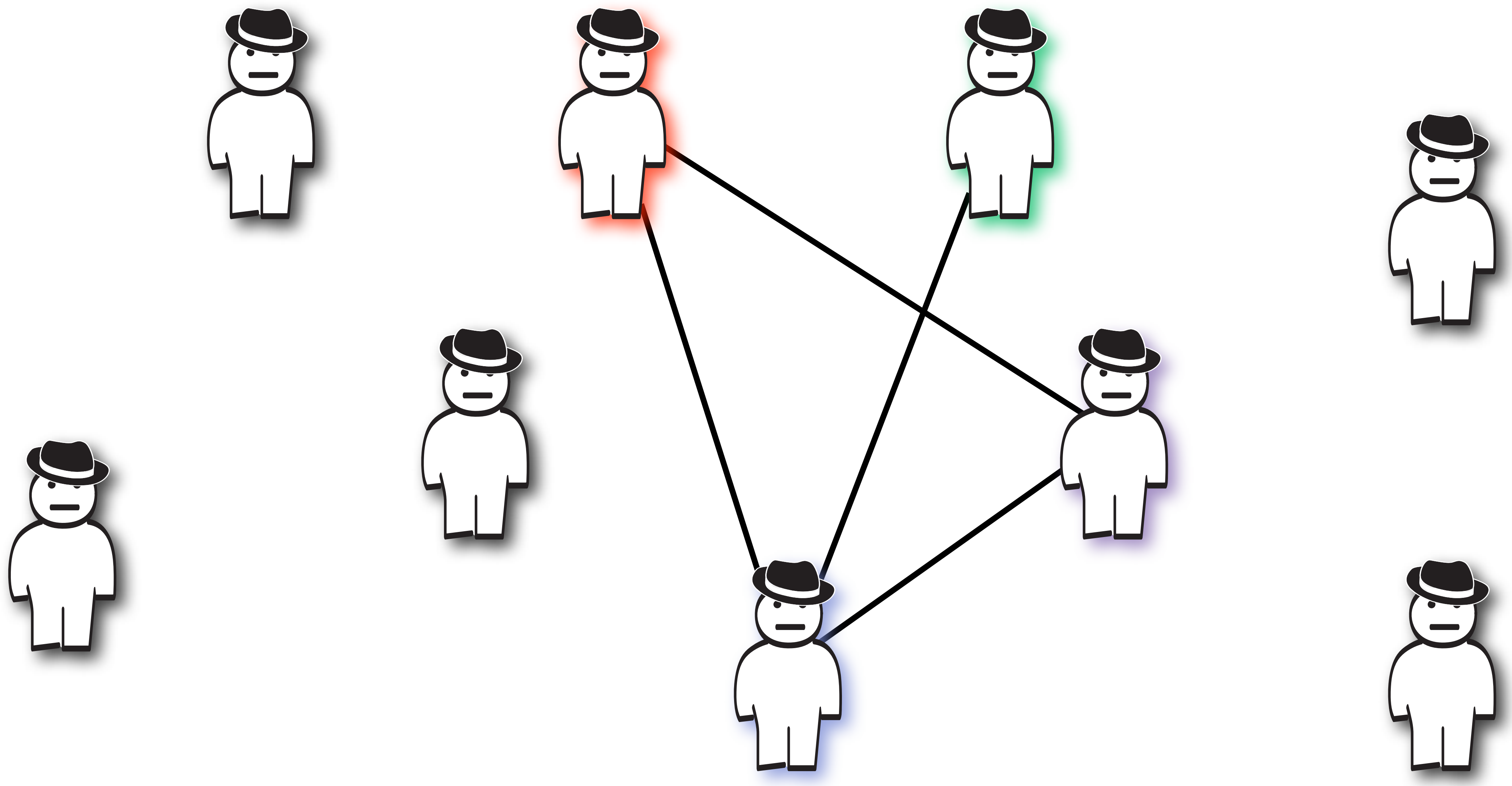
32

Social code networks



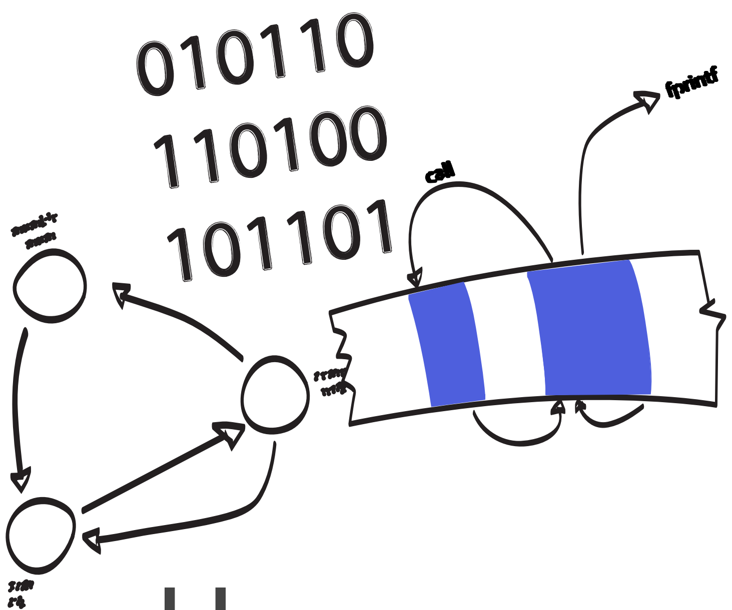
32

Social code networks



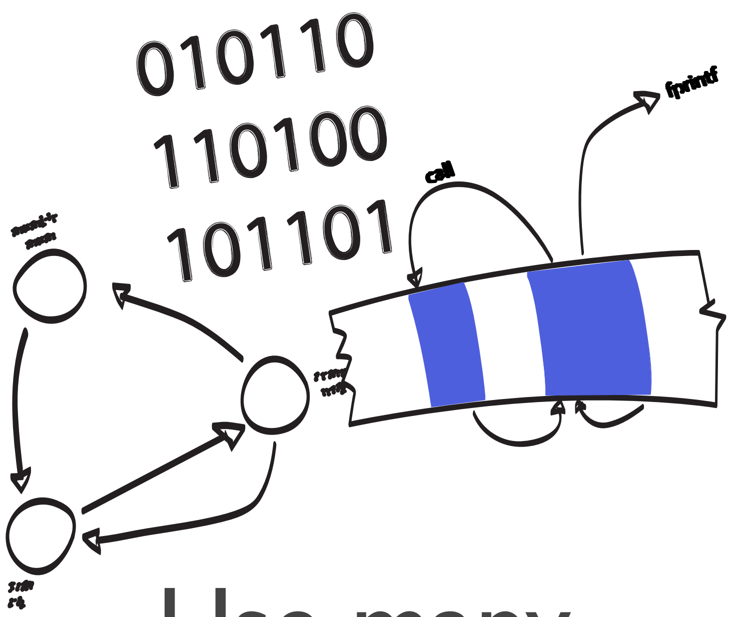
Summing up

Summing up

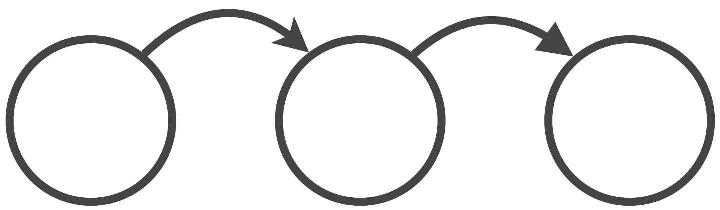


Use many
simple features

Summing up

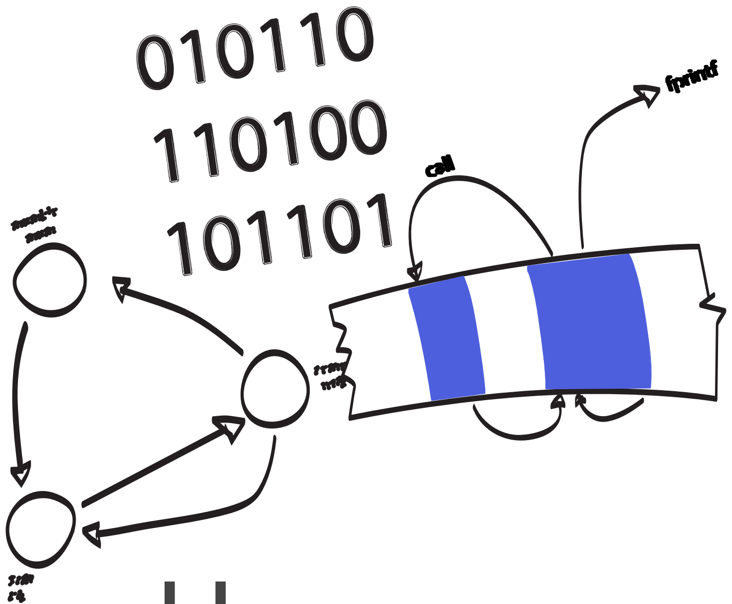


Use many
simple features

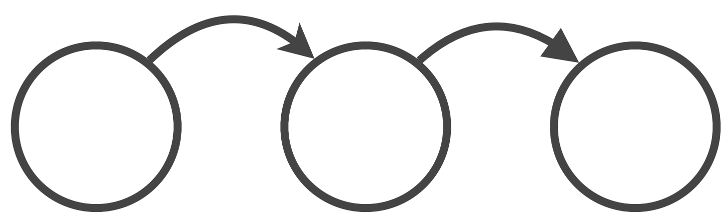


Take advantage
of structure

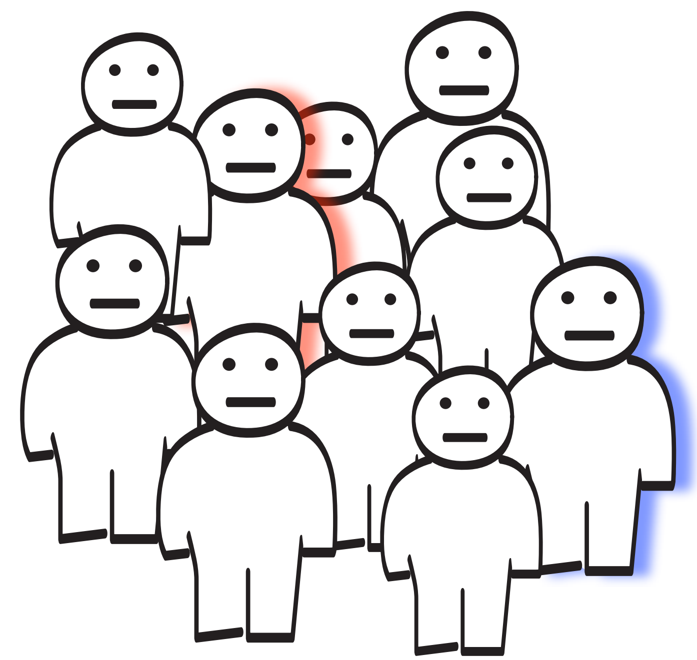
Summing up



Use many
simple features



Take advantage
of structure



questions