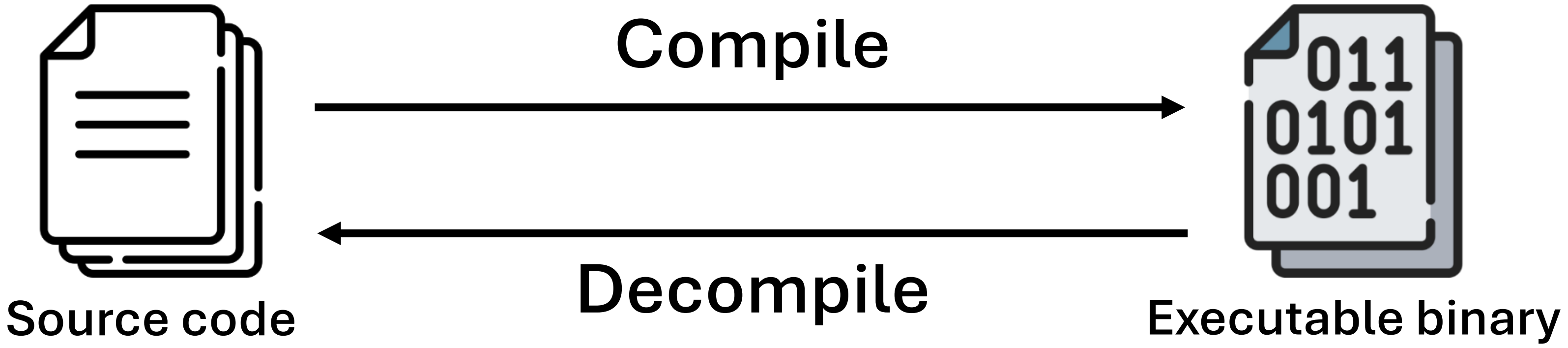
R2I: A Relative Readability Metric for Decompiled Code

Haeun Eom, Dohee Kim, Sori Lim, Hyungjoon Koo, Sungjae Hwang
Sungkyunkwan University

Background

▪ Decompiler

- A tool that performs the reversing process of compilation
  - Converting a low-level machine code into a high-level programming language

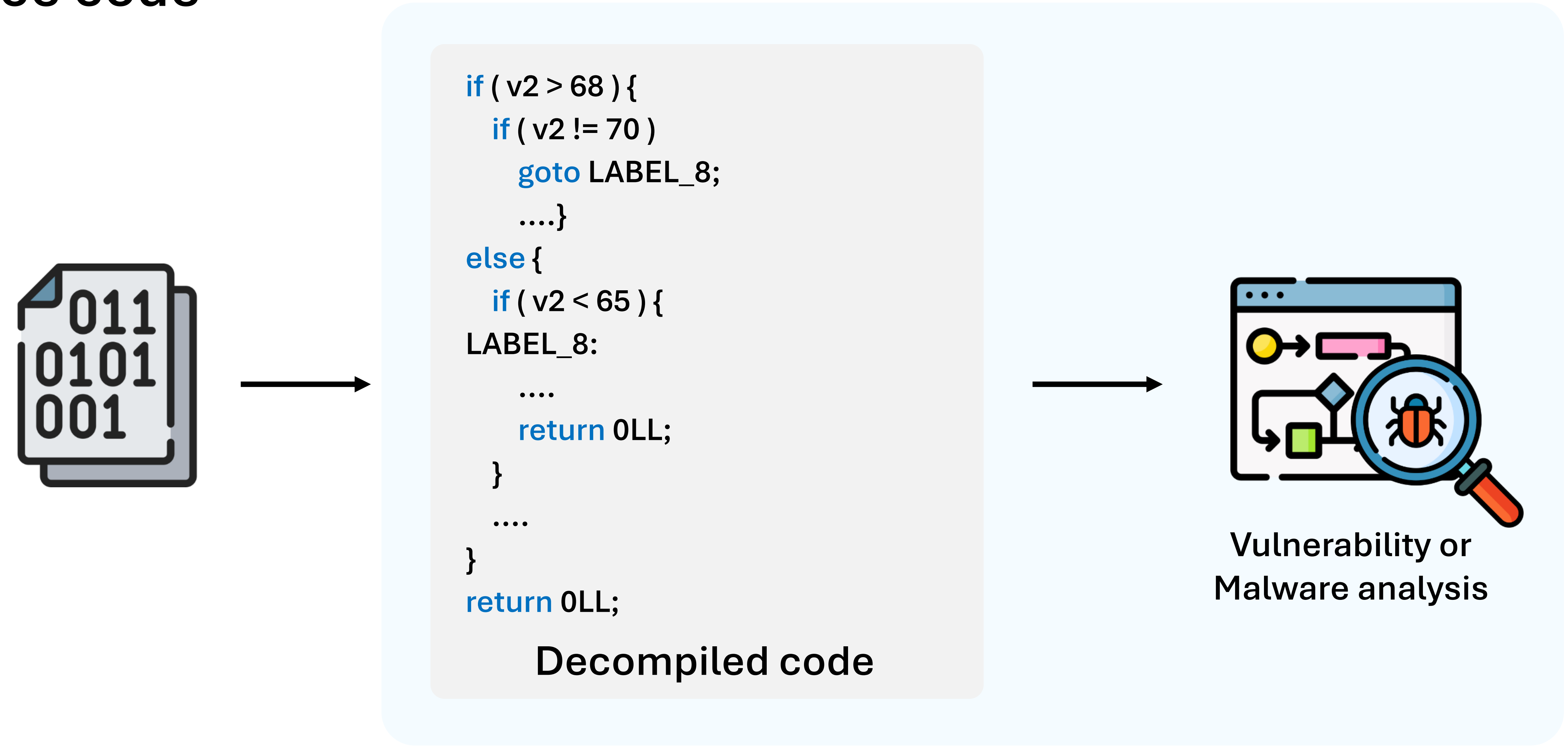


- Hex-Rays, Binary Ninja, Ghidra, Angr, Retdec, Radare2, …

Motivation

▪ Usage of decompiler

- Primarily utilized for analysis of contextual semantics of binaries without original source code



```
if ( v2 > 68 ) {
  if ( v2 != 70 )
    goto LABEL_8;
    ....}
else {
  if ( v2 < 65 ) {
LABEL_8:
    ....
    return 0LL;
  }
  ....
}
return 0LL;
```

Decompiled code

Vulnerability or Malware analysis

⇒ Readability of decompiled code is important for binary reversing

Motivation

Source code vs Decompiled code

```
void parse_long_options ( int argc, /*omitted*/, void
                  (*usage_func) (int), ...) {
  if (argc == 2 && (c = getopt_long(argc, argv, "+",long_options,
                                NULL)) != -1)
  {
    switch (c) {
      case 'h' :
        (*usage_func) (EXIT_SUCCESS);
        break;
      case 'v' : {
        va_list authors;
        va_start(authors, usage_func);
        version_etc_va(stdout, command_name, package,
                       version, authors);
        exit(0);
      }
      default :
        break;
    }
  }
  /*omitted*/
}
```

Source code

```
int64_t function_401b20(int64_t a1, /* omitted */, int64_t a6) {
  if ((char)v1 != 0) {
    /* omitted */
    __asm_movaps(v2);
  }
  int32_t v4 = function_404df0(a1, a2, &g3, (int64_t *)&g4, 0, a6);
  switch (v4) {
    default: {
      if (v4 == 118) {
        function_403c70((int64_t)g30, (int64_t)a3,
                        (int64_t)a4, a5, &v5, a6);
        exit(0);
      }
    }
    case -1: {
      (? > ?) ? 1 : 0;
    }
    case 104: {
      g27 = v3;
      return result2;
    }
}
```

Decompiled code

Motivation

▪ Source code vs Decompiled code

```
void parse_long_options ( int argc, /*omitted*/, void
                    (*usage_func) (int), ...) {
  if (argc == 2 && (c = getopt_long(argc, argv, "+",long_options,
                                    NULL)) != -1)
  {
    switch (c) {
      case 'h' :
        (*usage_func) (EXIT_SUCCESS);
        break;
      case 'v' : {
        va_list authors;
        va_start(authors, usage_func);
        version_etc_va(stdout, command_name, package,
                       version, authors);
        exit(0);
      }
      default :
        break;
    }
  }
  /*omitted*/
}
```

Source code

```
int64_t function_401b20(int64_t a1, /* omitted */, int64_t a6) {
  if ((char)v1 != 0) {
    /* omitted */
    __asm_movaps(v2);
  }
  int32_t v4 = function_404df0(a1, a2, &g3, (int64_t *)&g4, 0, a6);
  switch (v4) {
    default: {
      if (v4 == 118) {
        function_403c70((int64_t)g30, (int64_t)a3,
                        (int64_t)a4, a5, &v5, a6);
        exit(0);
      }
    }
    case -1: {
      (? > ?) ? 1 : 0;
    }
    case 104: {
      g27 = v3;
      return result2;
    }
  }
```

Decompiled code

Motivation

▪ Source code vs Decompiled code

```
void parse_long_options ( int argc, /*omitted*/, void
                  (*usage_func) (int), ...) {
  if (argc == 2 && (c = getopt_long(argc, argv, "+",long_options,
                                  NULL)) != -1)
  {
    switch (c) {
      case 'h' :
        (*usage_func) (EXIT_SUCCESS);
        break;
      case 'v' : {
        va_list authors;
        va_start(authors, usage_func);
        version_etc_va(stdout, command_name, package,
                       version, authors);
        exit(0);
      }
      default :
        break;
    }
  }
  /*omitted*/
}
```

Source code

```
int64_t function_401b20(int64_t a1, /* omitted */, int64_t a6) {
  if ((char)v1 != 0) {
    /* omitted */
    __asm_movaps(v2);
  }
  int32_t v4 = function_404df0(a1, a2, &g3, (int64_t *)&g4, 0, a6);
  switch (v4) {
    default: {
      if (v4 == 118) {
        function_403c70((int64_t)g30, (int64_t)a3,
                        (int64_t)a4, a5, &v5, a6);
        exit(0);
      }
    }
    case -1: {
      (? > ?) ? 1 : 0;
    }
    case 104: {
      g27 = v3;
      return result2;
    }
}
```

Decompiled code

Motivation

- ## Decompiler outputs

```
int sub_401650() {
    /*omitted*/
    if (v1 == v2) {
        if ((v3[0] &223) == 85) {
            /*omitted*/
        }
        else {
            if (v4 ==71 && (v3[1] &223) == 66
            && v3[2] ==49 && v3[3] ==56 && v3[4] ==48
            && v3[5] ==51 && v3[6] ==48 && v3[7] ==0) {
                v2 = ((v2) !=96 ? &g_403a0a : 4209165);
            }
        }
    }
    if (...) {
        return ((unsigned int) v5 != 9 ? "" : "\"");
    }
    /*omitted*/
}
```

Hex-Rays

```
uint64_t fcn_00401650 (int64_t arg1, int64_t arg2) {
   /*omitted*/
label_0:
        if (dl != 0x55)
            goto label_1;
        /*omitted*/
label_1:
    if (dl ==0x47) {
        edx = ((rax +1));
        edx &= 0xffffffdf;
        if (dl != 0x42)
            goto label_2;
        /*omitted*/
        if (*(rbx) != 0x60)
            rbx = rax;
label_2:
    if (r12d != 9)
        rbx = rax;
     /*omitted*/
}
```

Radare2

Motivation

▪ Decompiler outputs

```
int sub_401650() {
    /*omitted*/
    if (v1 == v2) {
        if ((v3[0] &223) == 85) {
            /*omitted*/
        }
        else {
            if (v4 ==71 && (v3[1] &223) == 66
            && v3[2] ==49 && v3[3] ==56 && v3[4] ==48
            && v3[5] ==51 && v3[6] ==48 && v3[7] ==0) {
                v2 = ((v2) !=96 ? &g_403a0a : 4209165);
            }
        }
    }
    if (...) {
        return ((unsigned int) v5 != 9 ? "" : "\"");
    }
    /*omitted*/
}
```

Hex-Rays

```
uint64_t fcn_00401650 (int64_t arg1, int64_t arg2) {
    /*omitted*/
label_0:
        if (dl != 0x55)
            goto label_1;
        /*omitted*/
label_1:
    if (dl ==0x47) {
        edx = ((rax +1));
        edx &= 0xffffffdf;
        if (dl != 0x42)
            goto label_2;
        /*omitted*/
        if (*(rbx) != 0x60)
            rbx = rax;
label_2:
    if (r12d != 9)
        rbx = rax;
     /*omitted*/
}
```

Radare2

Motivation

▪ Decompiler outputs

```
int sub_401650() {
    /*omitted*/
    if (v1 == v2) {
        if ((v3[0] &223) == 85) {
            /*omitted*/
        }
        else {
            if (v4 ==71 && (v3[1] &
            && v3[2] ==49 && v3[3
            && v3[5] ==51 && v3[6
                v2 = ((v2) !=96 ? &g_
            }
        }
    }
    if (...) {
        return ((unsigned int) v5 != 9 ? """ : "\"");
    }
    /*omitted*/
}
```

Output 1

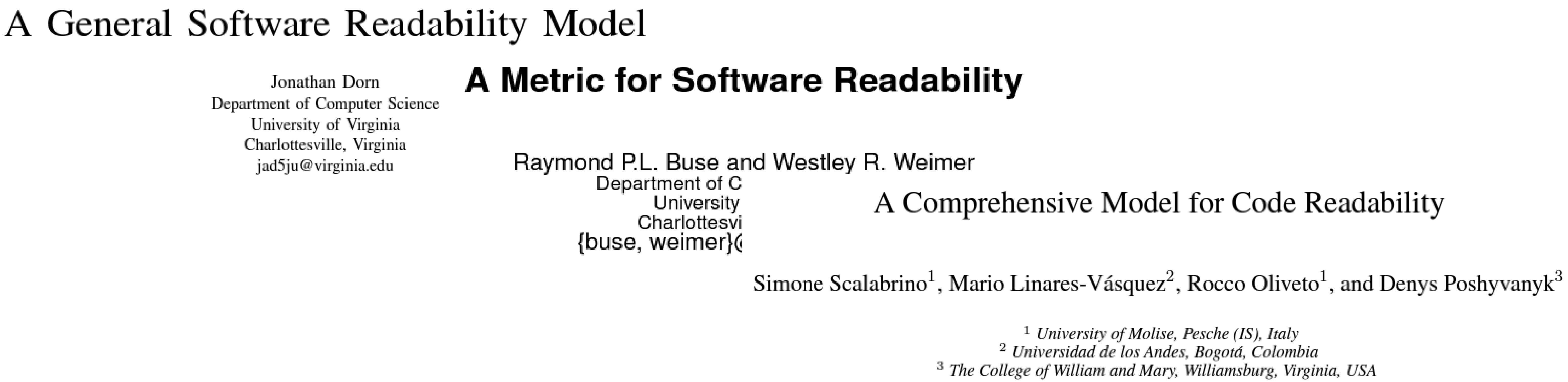
```
uint64_t fcn_00401650 (int64_t arg1, int64_t arg2) {
  /*omitted*/
label_0:
        if (dl != 0x55)
            goto label_1;

            goto label_2;
        /*omitted*/
        if (*(rbx) != 0x60)
            rbx = rax;
label_2:
    if (r12d != 9)
        rbx = rax;
     /*omitted*/
}
```

Output 2

No specific readability metrics for decompiled code

Related work

▪ Source code readability metrics



A General Software Readability Model

Jonathan Dorn
Department of Computer Science
University of Virginia
Charlottesville, Virginia
jad5ju@virginia.edu

A Metric for Software Readability

Raymond P.L. Buse and Westley R. Weimer
Department of C
University
Charlottesvi
{buse, weimer}@

A Comprehensive Model for Code Readability

Simone Scalabrino[1], Mario Linares-Vásquez[2], Rocco Oliveto[1], and Denys Poshyvanyk[3]

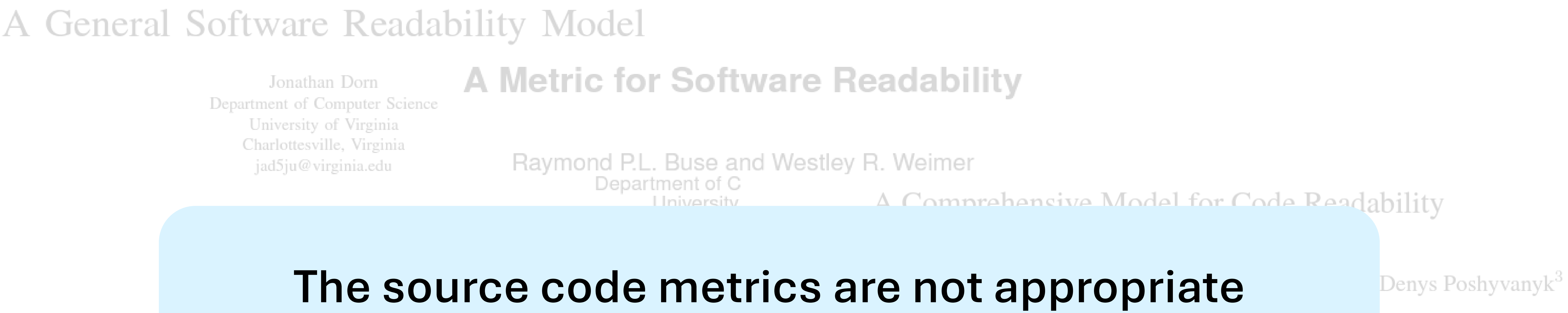
[1] *University of Molise, Pesche (IS), Italy*
[2] *Universidad de los Andes, Bogotá, Colombia*
[3] *The College of William and Mary, Williamsburg, Virginia, USA*

⇒ Numerous semantic features specific to source code

(e.g. Identifier length, Comments, Identifiers meaning, Data type, etc.)

Related work

▪ Source code readability metrics



A General Software Readability Model

Jonathan Dorn
Department of Computer Science
University of Virginia
Charlottesville, Virginia
jad5ju@virginia.edu

A Metric for Software Readability

Raymond P.L. Buse and Westley R. Weimer
Department of C
University

A Comprehensive Model for Code Readability

Denys Poshyvanyk[3]

The source code metrics are not appropriate
for the readability of decompiled code

⇒ Numerous semantic features specific to source code

(e.g. Identifier length, Comments, Identifiers meaning, Data type, etc.)

Challenges

- **Computing an absolute metric is not feasible**
  - No original code is available, having no ground truth to measure readability

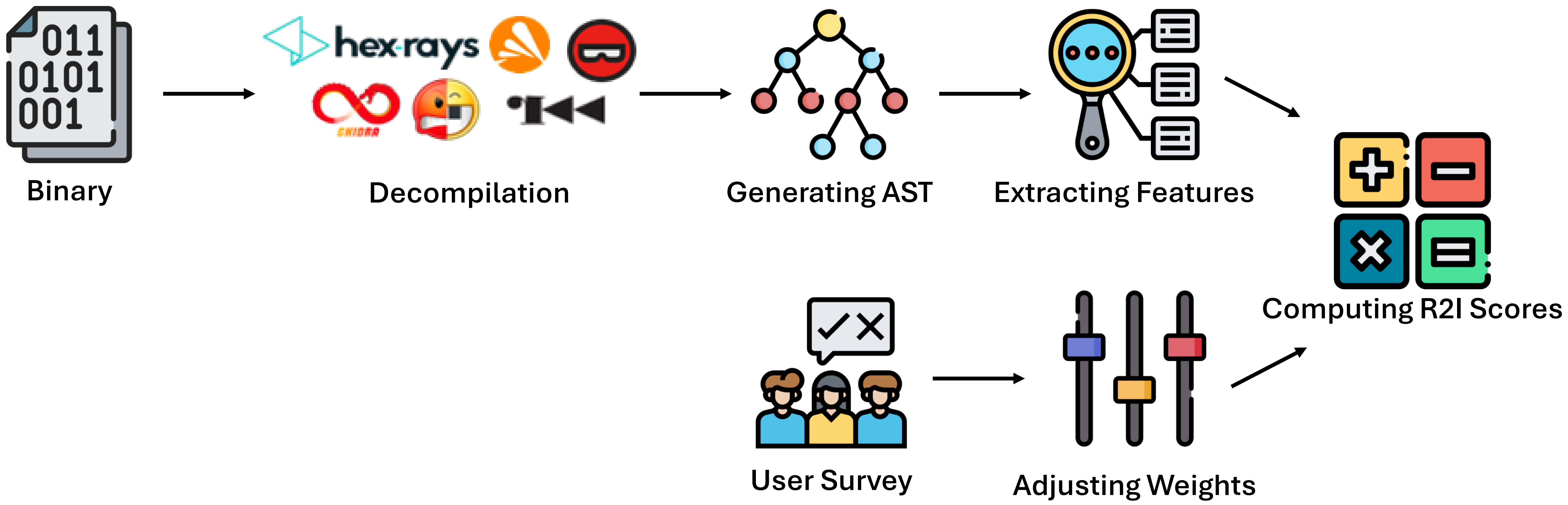
- **Decompiled-code-oriented features have been under-explored**
  - Existing readability features are for source code

- **Automatic feature extraction is challenging**
  - Various and frequent grammatical errors in decompiled code

R2I : Relative Readability Index

Overview

- First readability metric tailored to decompiled code



Binary

Decompilation

Generating AST

Extracting Features

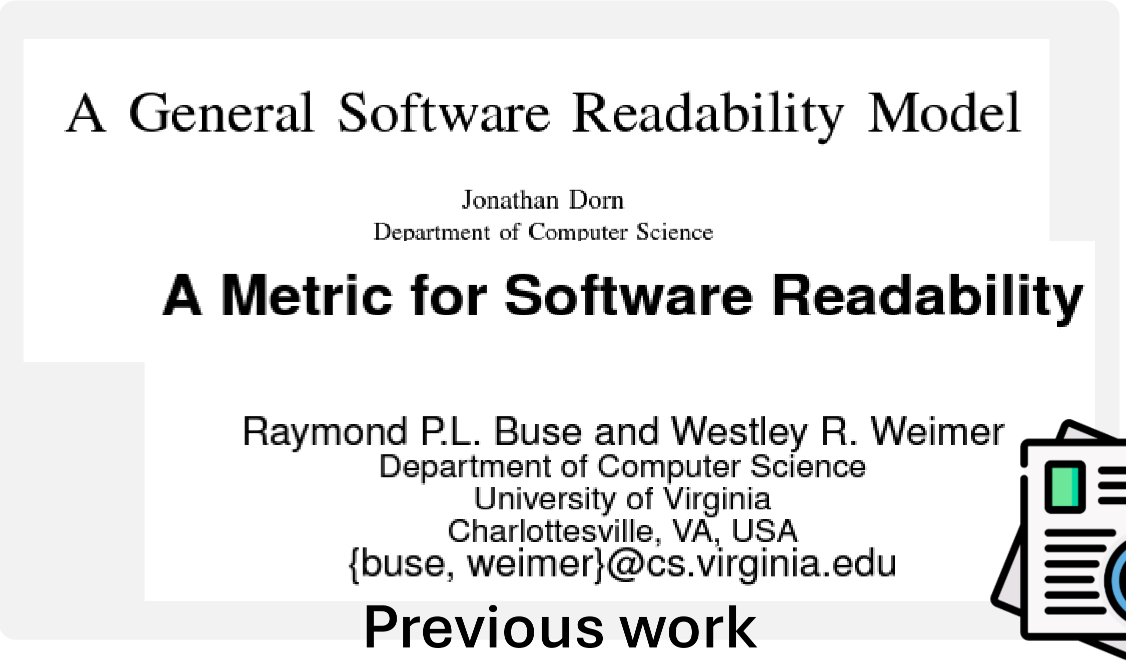
Computing R2I Scores

User Survey

Adjusting Weights

R2I – Feature Definition

▪ Decompiled code features - criteria



A General Software Readability Model

Jonathan Dorn
Department of Computer Science

A Metric for Software Readability

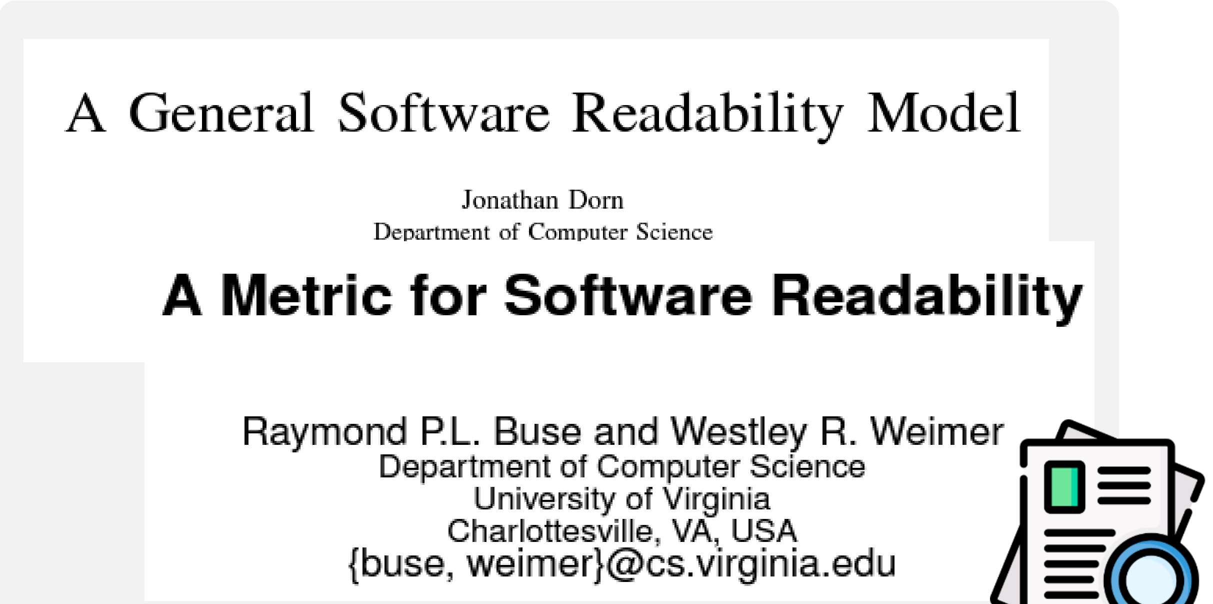
Raymond P.L. Buse and Westley R. Weimer
Department of Computer Science
University of Virginia
Charlottesville, VA, USA
{buse, weimer}@cs.virginia.edu

Previous work

- 6 Source code readability metrics
- 9 Readability-affecting factors
- 4 Decompiler-enhancing efforts

R2I – Feature Definition

- **Decompiled code features - criteria**



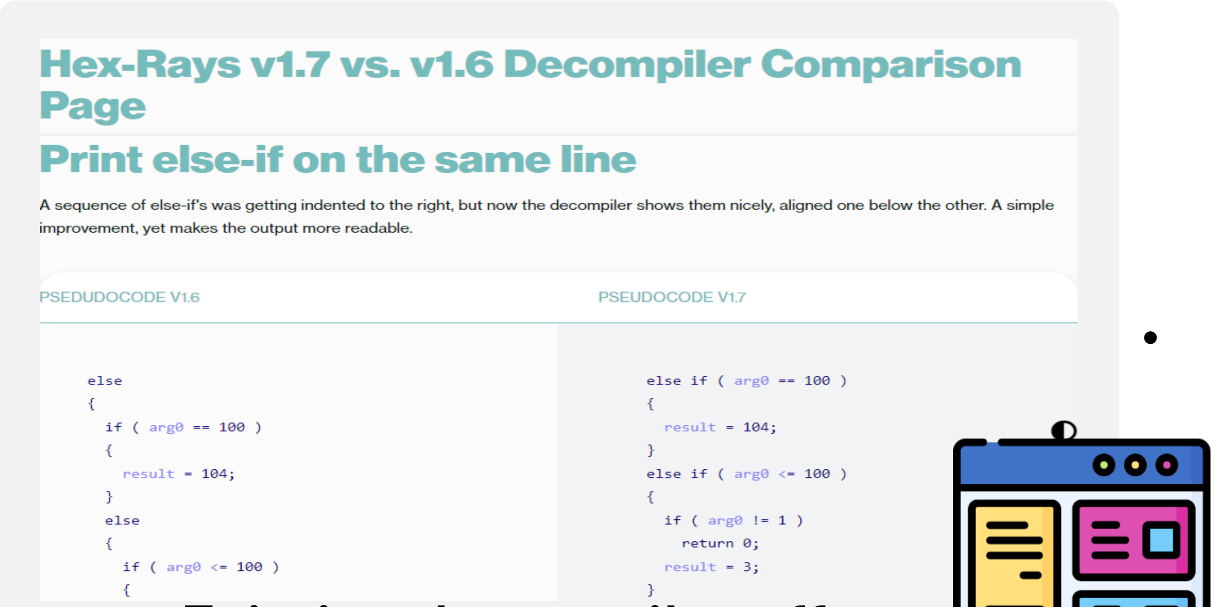
A General Software Readability Model

Jonathan Dorn
Department of Computer Science

A Metric for Software Readability

Raymond P.L. Buse and Westley R. Weimer
Department of Computer Science
University of Virginia
Charlottesville, VA, USA
{buse, weimer}@cs.virginia.edu

Previous work



Hex-Rays v1.7 vs. v1.6 Decompiler Comparison Page

Print else-if on the same line

A sequence of else-if's was getting indented to the right, but now the decompiler shows them nicely, aligned one below the other. A simple improvement, yet makes the output more readable.

PSEDUDOCODE V1.6

```
else
{
  if ( arg0 == 100 )
  {
    result = 104;
  }
  else
  {
    if ( arg0 <= 100 )
    {
```

PSEUDOCODE V1.7

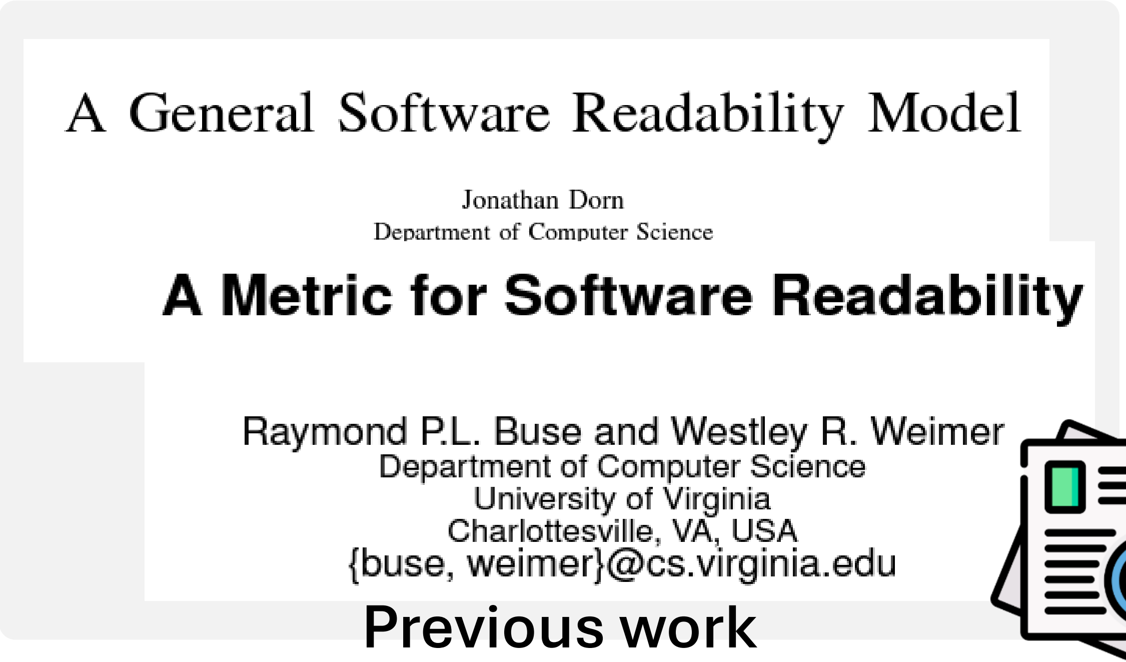
```
else if ( arg0 == 100 )
{
  result = 104;
}
else if ( arg0 <= 100 )
{
  if ( arg0 != 1 )
    return 0;
  result = 3;
}
```

Existing decompiler efforts

- 174 changelogs

R2I – Feature Definition

▪ Decompiled code features - criteria



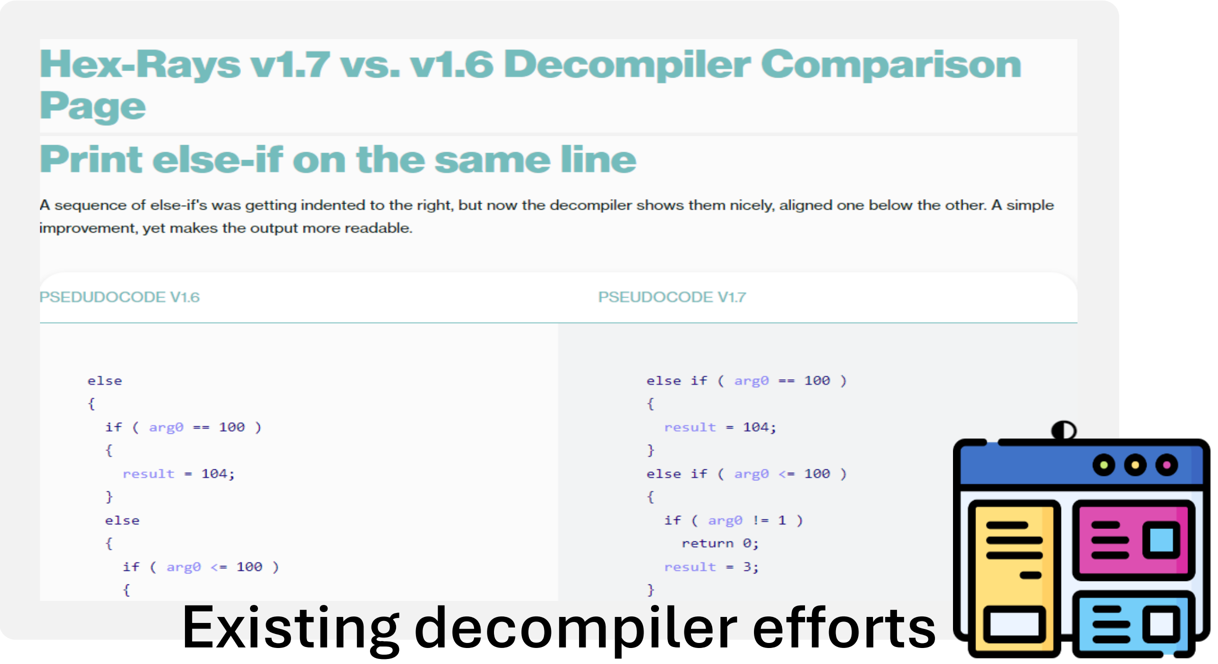
A General Software Readability Model

Jonathan Dorn
Department of Computer Science

A Metric for Software Readability

Raymond P.L. Buse and Westley R. Weimer
Department of Computer Science
University of Virginia
Charlottesville, VA, USA
{buse, weimer}@cs.virginia.edu

Previous work



Hex-Rays v1.7 vs. v1.6 Decompiler Comparison Page

Print else-if on the same line

A sequence of else-if's was getting indented to the right, but now the decompiler shows them nicely, aligned one below the other. A simple improvement, yet makes the output more readable.

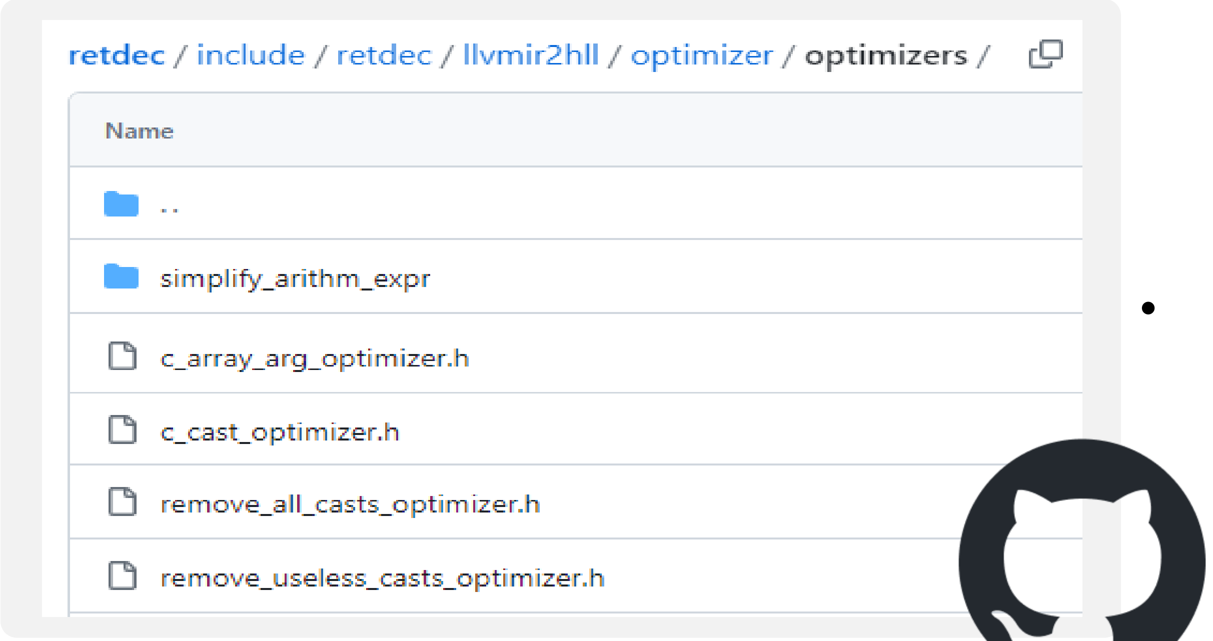
PSEDUDOCODE V1.6

```
else
{
  if ( arg0 == 100 )
  {
    result = 104;
  }
  else
  {
    if ( arg0 <= 100 )
    {
```

PSEUDOCODE V1.7

```
else if ( arg0 == 100 )
{
  result = 104;
}
else if ( arg0 <= 100 )
{
  if ( arg0 != 1 )
    return 0;
  result = 3;
}
```

Existing decompiler efforts



retdec / include / retdec / llvmir2hll / optimizer / optimizers /

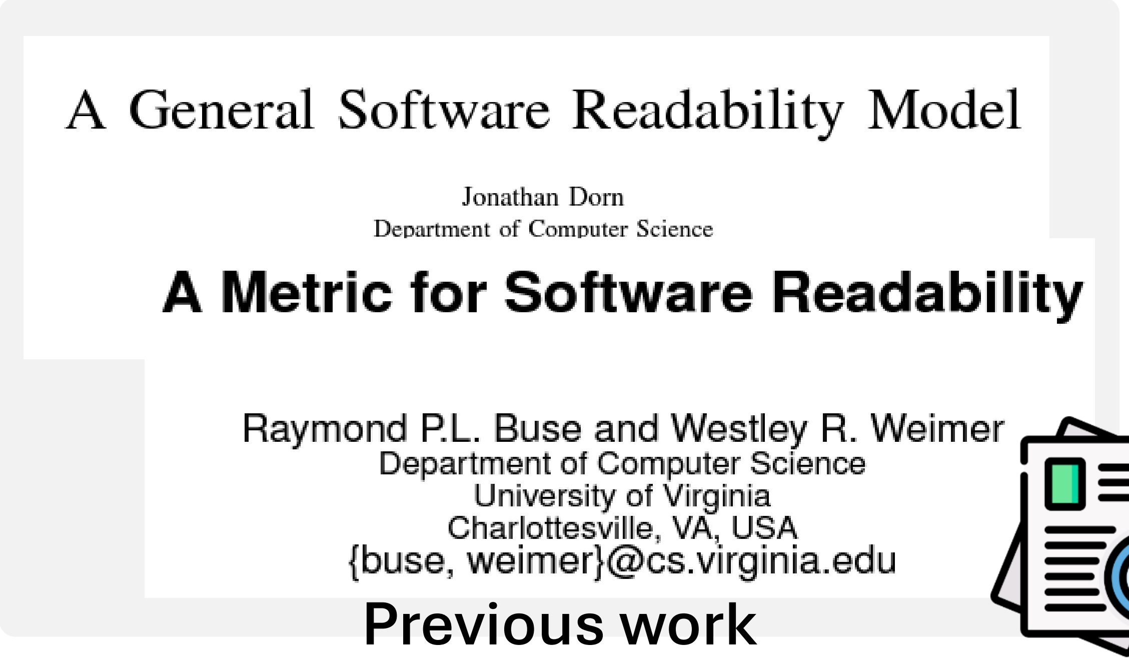
Name
..
simplify_arithm_expr
c_array_arg_optimizer.h
c_cast_optimizer.h
remove_all_casts_optimizer.h
remove_useless_casts_optimizer.h

- 64 optimizers

Optimization modules

R2I – Feature Definition

Decompiled code features - criteria



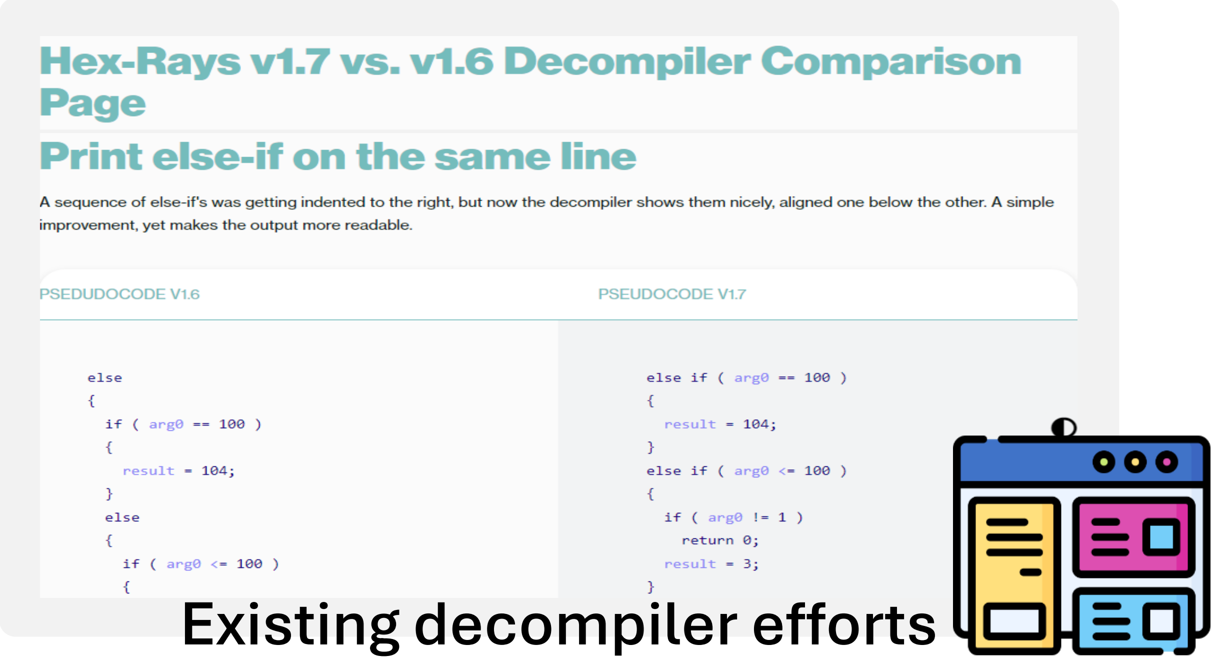
A General Software Readability Model

Jonathan Dorn
Department of Computer Science

A Metric for Software Readability

Raymond P.L. Buse and Westley R. Weimer
Department of Computer Science
University of Virginia
Charlottesville, VA, USA
{buse, weimer}@cs.virginia.edu

Previous work



Hex-Rays v1.7 vs. v1.6 Decompiler Comparison Page

Print else-if on the same line

A sequence of else-if's was getting indented to the right, but now the decompiler shows them nicely, aligned one below the other. A simple improvement, yet makes the output more readable.

PSEDUDOCODE V1.6

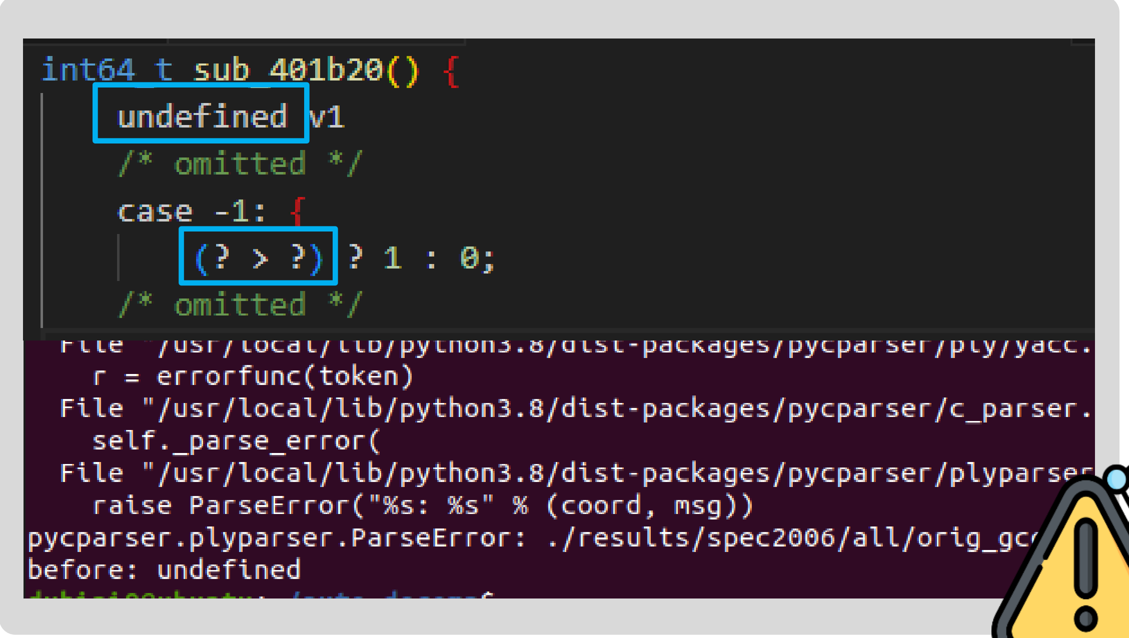
```
else
{
  if ( arg0 == 100 )
  {
    result = 104;
  }
  else
  {
    if ( arg0 <= 100 )
    {
```

PSEUDOCODE V1.7

```
else if ( arg0 == 100 )
{
  result = 104;
}
else if ( arg0 <= 100 )
{
  if ( arg0 != 1 )
    return 0;
  result = 3;
}
```

Existing decompiler efforts

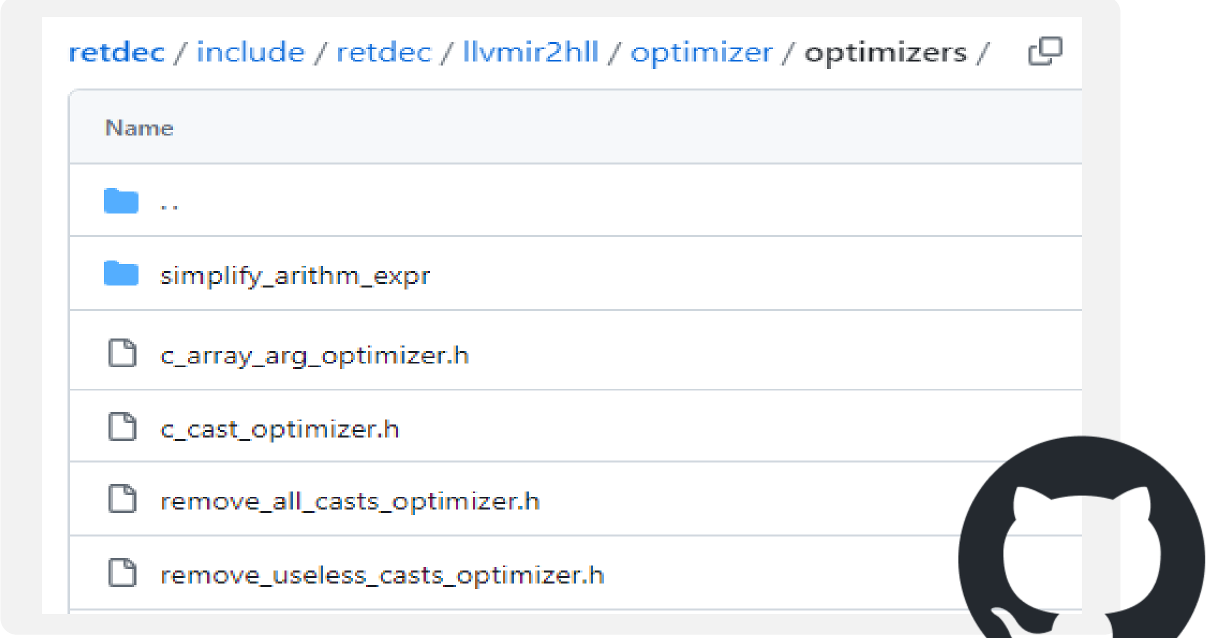
- 68,464 functions



```
int64_t sub_401b20() {
    undefined v1
    /* omitted */
    case -1: {
        (? > ?) ? 1 : 0;
    /* omitted */
```

```
  File "/usr/local/lib/python3.8/dist-packages/pycparser/ply/yacc.
    r = errorfunc(token)
  File "/usr/local/lib/python3.8/dist-packages/pycparser/c_parser.
    self._parse_error(
  File "/usr/local/lib/python3.8/dist-packages/pycparser/plyparse
    raise ParseError("%s: %s" % (coord, msg))
pycparser.plyparser.ParseError: ./results/spec2006/all/orig_gc
before: undefined
```

Syntactic errors



retdec / include / retdec / llvmir2hll / optimizer / optimizers /

Name
..
simplify_arithm_expr
c_array_arg_optimizer.h
c_cast_optimizer.h
remove_all_casts_optimizer.h
remove_useless_casts_optimizer.h

Optimization modules

R2I – Feature Definition

- **31 decompiled code features**

Class	Feature
Code Quality	# of array detections
	# of operators
	# of comma operators in conditions
	# of goto statements
	# of inline assembly
	# of missing conditions
	# of nested casting operators
	# of references/dereferences
	# of unnecessary goto labels
	# of variables
User Preference	Ratio of conditional statements
	Ratio of loop statements
	Ratio of !strcmp in conditions
Conflicting Features	Max # of conditions in if statements
	Max # of nested if statements
	Max length of a line
	Line of code

Class	Feature
Erroneous Syntax	# of multiple types
	# of invalid goto statements
	# of invalid do-while loops
	# of invalid function calls
	# of remaining IRs
	# of unimplemented parts
	# of unknown expressions
	# of invalid argument
	# of unknown operators
General Features	# of tokens
	# of conditions
	# of loops
	# of assignments
	Max # of nested loop statements

R2I – Feature Definition

- **31 decompiled code features**

Class	Feature
Code Quality	# of array detections
	# of operators
	# of comma operators in conditions
	# of goto statements
	# of inline assembly
	# of missing conditions
	# of nested casting operators
	# of references/dereferences
	# of unnecessary goto labels
	# of variables
User Preference	Ratio of conditional statements
	Ratio of loop statements
	Ratio of !strcmp in conditions
Conflicting Features	Max # of conditions in if statements
	Max # of nested if statements
	Max length of a line
	Line of code

Class	Feature
Erroneous Syntax	# of multiple types
	# of invalid goto statements
	# of invalid do-while loops
	# of invalid function calls
	# of remaining IRs
	# of unimplemented parts
	# of unknown expressions
	# of invalid argument
	# of unknown operators
General Features	# of tokens
	# of conditions
	# of loops
	# of assignments
	Max # of nested loop statements

R2I – Feature Definition

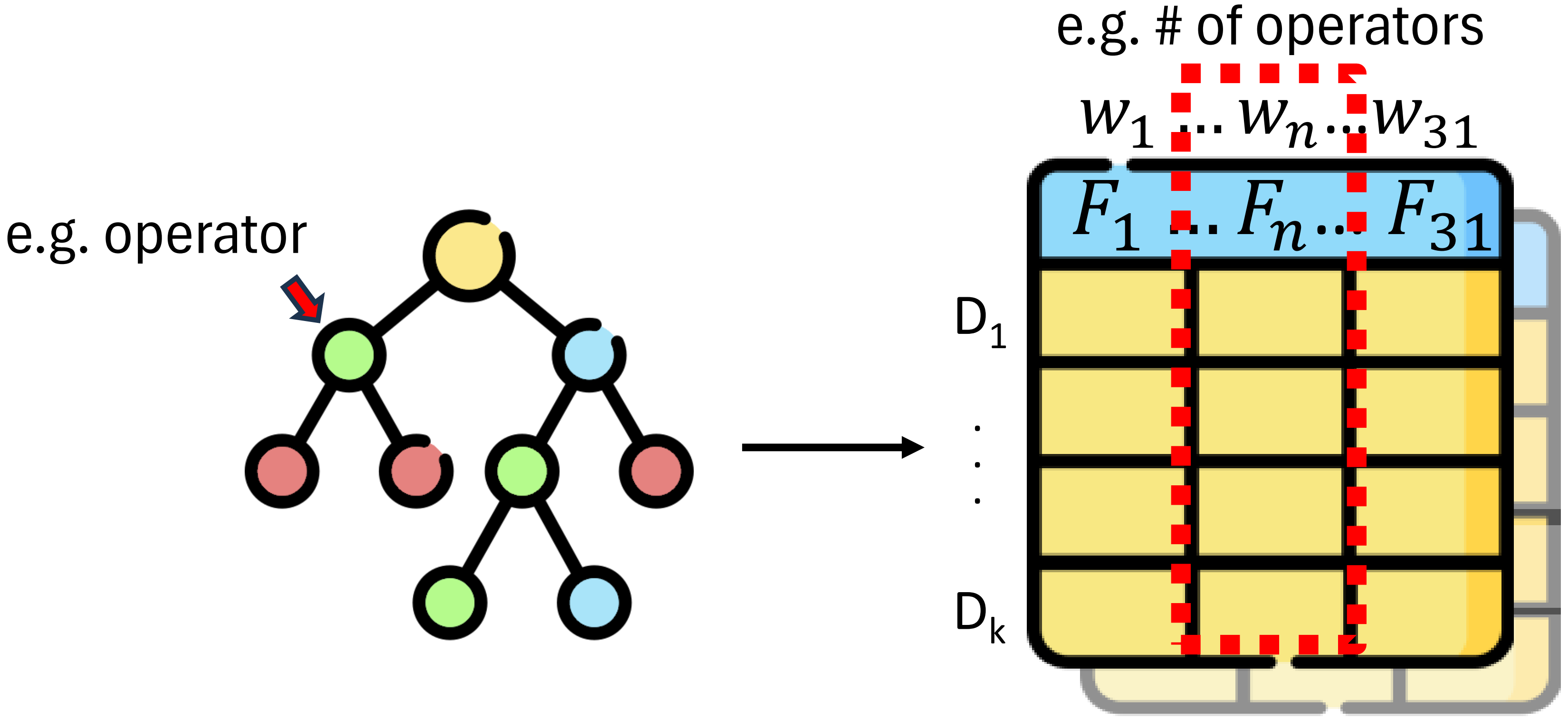
Erroneous syntax

- Various syntax errors in C-like decompiled code outputs
- Error correction using custom headers and regular expressions

Error Category	Error Type	Example	Correction
Invalid Data Types	Declarations	unsigned int | char v0;	undefined v0;
		undefined v1;	typedef int undefined;
		(_UNKNOWN *) v19;	typedef void _UNKNOWN;
		code **ppcVar1;	typedef int code;
Invalid Expressions	Structures	LAB_004c8dba: }	INVAL_LAB;
		do{ .. } ..}while(..)	INVAL_DOWHILE;
	Identifiers	void(*0x401350)()();	INVAL_FUNCALL();
	Eccentricities	Conv(16 -> 128, d1);	INVALID_IR;
		x = /*x = unimplemented { }*/;	x = UNIMPL;
	Expressions	if(...)	if(unknown)
		(? > ?) ? 1 : 0;	(unknown) ? 1 : 0;
		? = fp_stack[0]	(unknown) = fp_sp_stack[0]
		setjmp({(struct{ })	setjmp(INVAL_FORM)
	Operators	if (ebp overflow 0)	if(UNKNOWN_OP)

R2I – R2I Computation

Features extraction



e.g. # of operators

$w_1 \ldots w_n \ldots w_{31}$

$F_1 \ldots F_n \ldots F_{31}$

e.g. operator

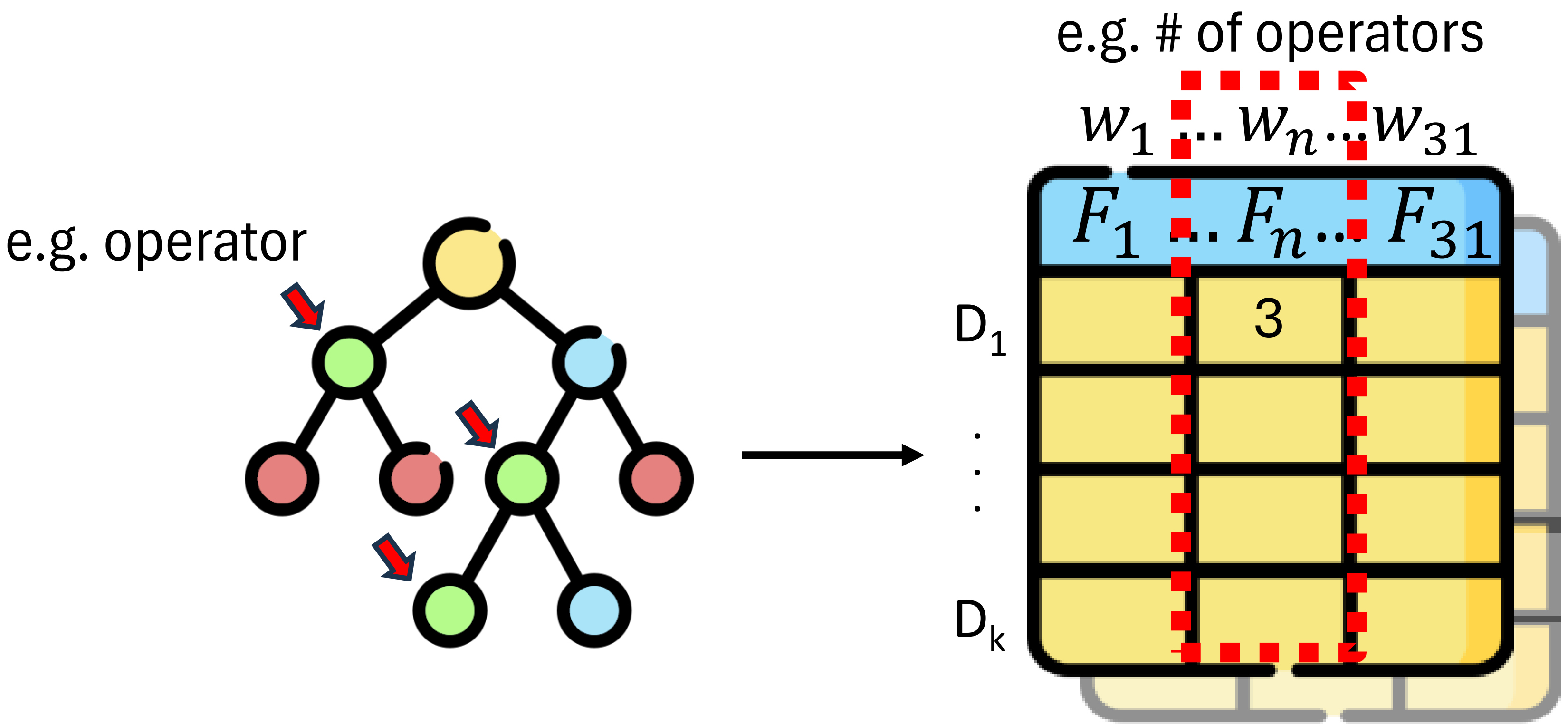
D_1

D_k

- F - feature
- D - decompiler
- f - occurrence of the feature
- w - weight of the feature

R2I – R2I Computation

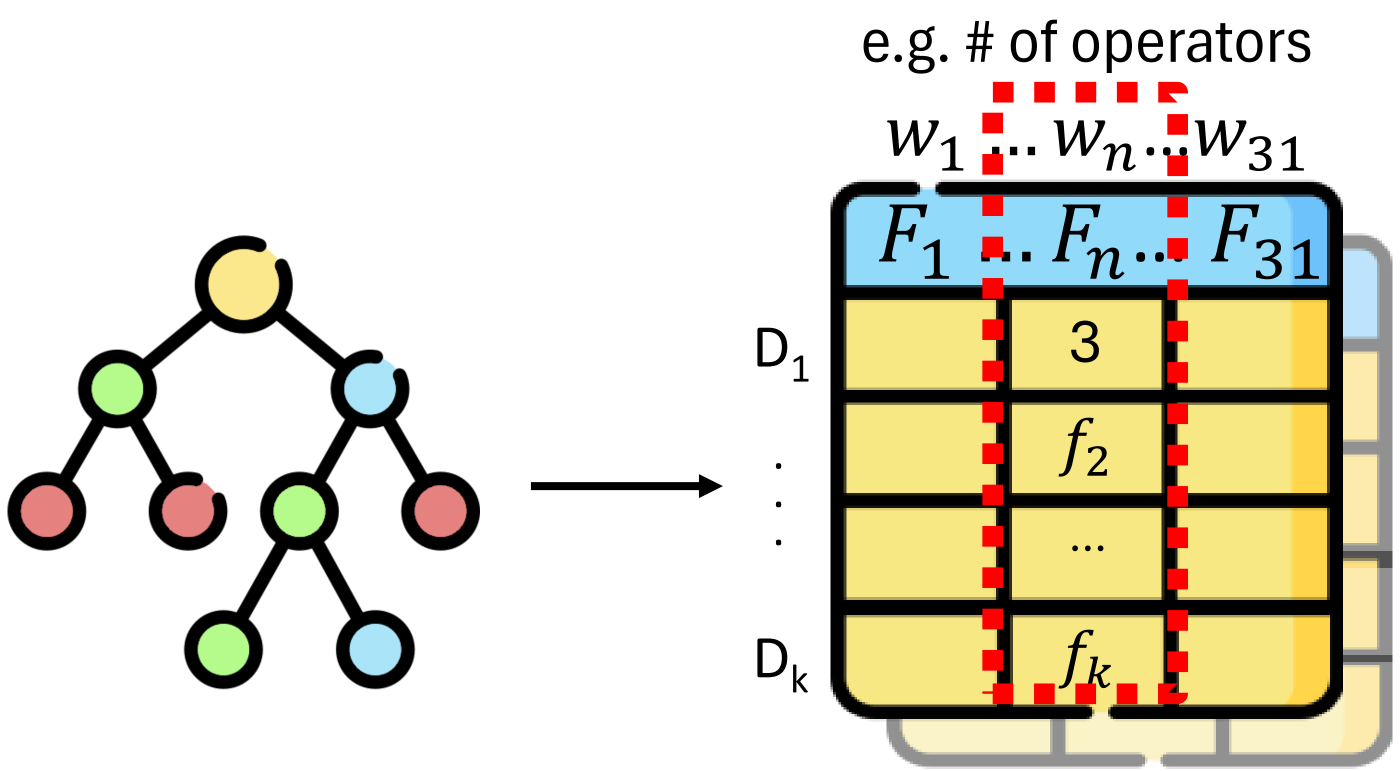
Features extraction



- F - feature
- D - decompiler
- f - occurrence of the feature
- w - weight of the feature

R2I – R2I Computation

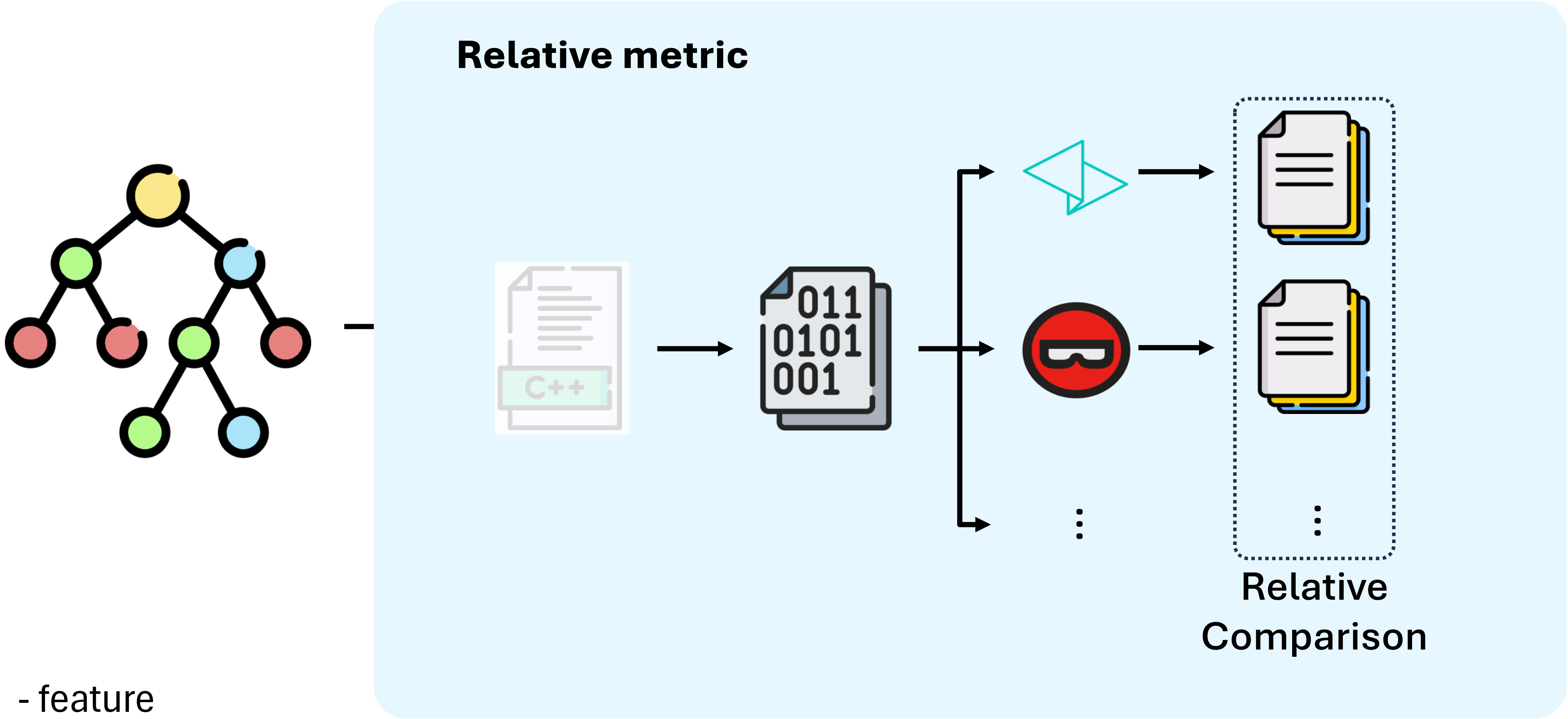
Features extraction



- F - feature
- D - decompiler
- f - occurrence of the feature
- w - weight of the feature

R2I – R2I Computation

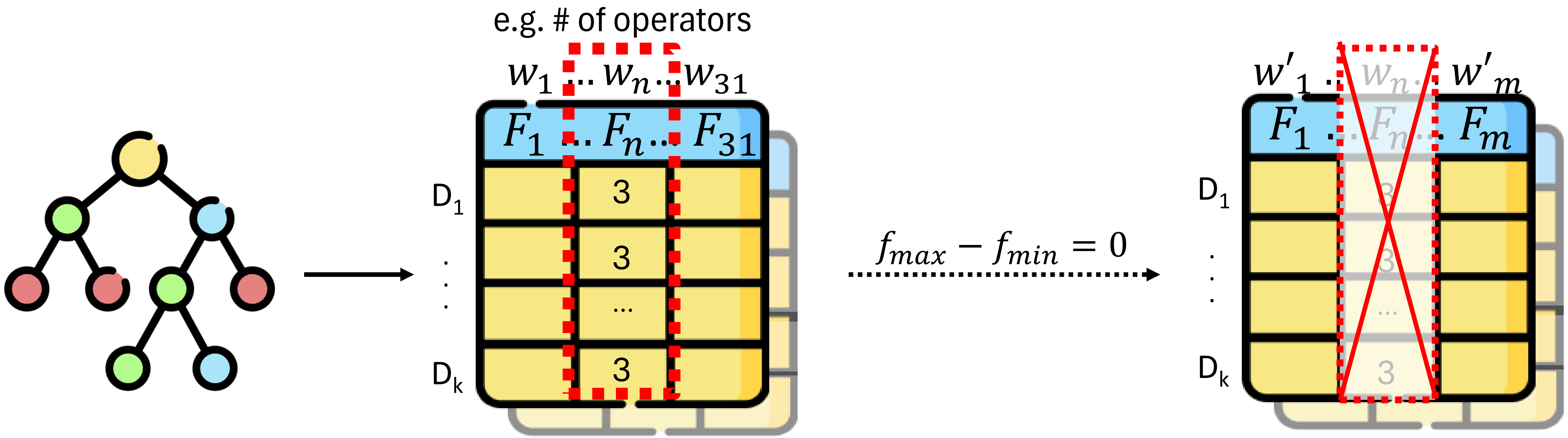
▪ Features extraction



- F - feature
- D - decompiler
- f - occurrence of the feature
- w - weight of the feature

R2I – R2I Computation

Features extraction

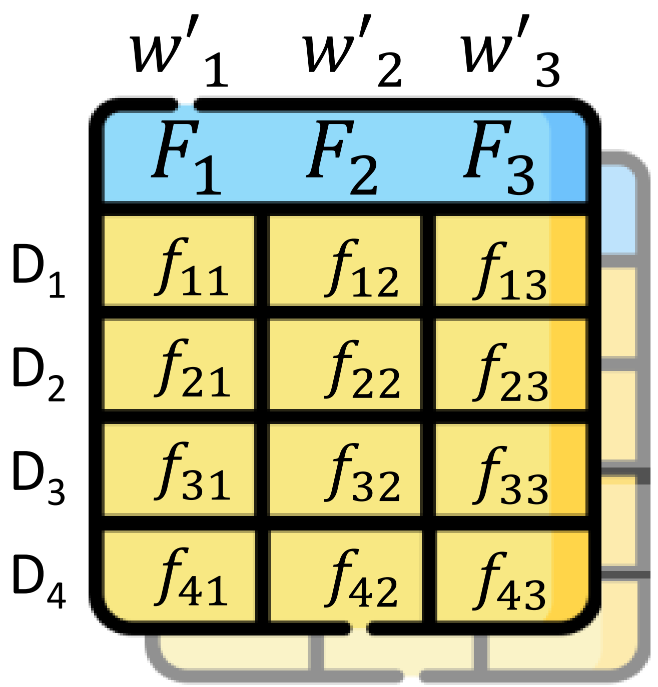


e.g. # of operators
w_1 ... w_n ... w_{31}
F_1 ... F_n ... F_{31}
D$_1$
D$_k$
3
3
...
3
$f_{max} - f_{min} = 0$
w'_1 ... w_n ... w'_m
F_1 ... F_n ... F_m
D$_1$
D$_k$
3
3
...
3

- F - feature
- D - decompiler
- f - occurrence of the feature
- w - weight of the feature

R2I – R2I Computation

▪ Index computation



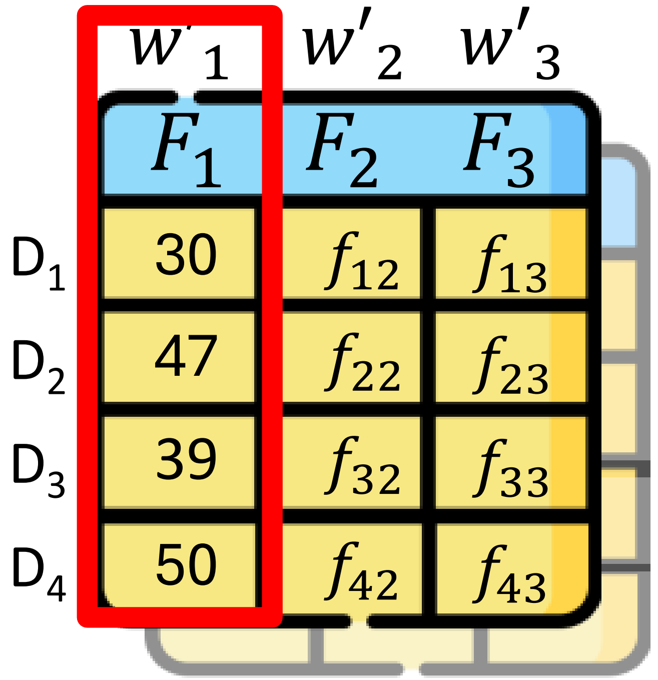
	w'_1	w'_2	w'_3
	F_1	F_2	F_3
D_1	f_{11}	f_{12}	f_{13}
D_2	f_{21}	f_{22}	f_{23}
D_3	f_{31}	f_{32}	f_{33}
D_4	f_{41}	f_{42}	f_{43}

- F - feature
- D - decompiler
- f - occurrence of the feature
- w - weight of the feature

R2I – R2I Computation

Index computation

e.g. # of tokens



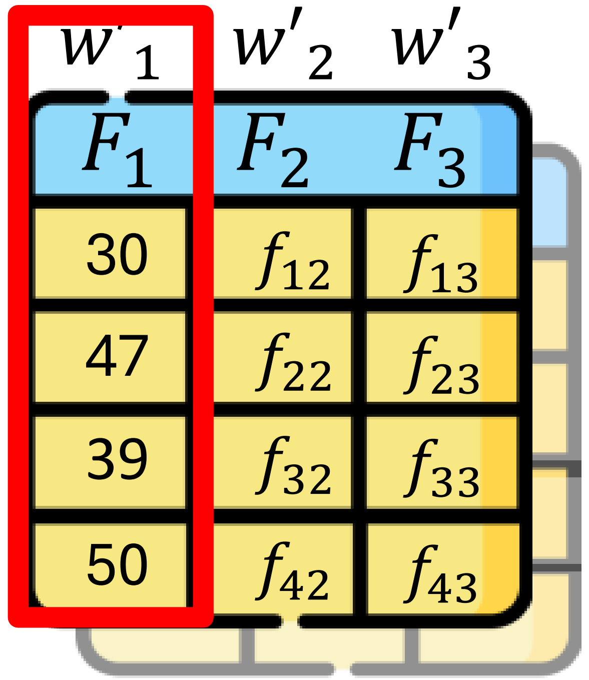
	w'_1	w'_2	w'_3
	F_1	F_2	F_3
D_1	30	f_{12}	f_{13}
D_2	47	f_{22}	f_{23}
D_3	39	f_{32}	f_{33}
D_4	50	f_{42}	f_{43}

- F - feature
- D - decompiler
- f - occurrence of the feature
- w - weight of the feature

R2I – R2I Computation

Index computation

e.g. # of tokens



	w'_1	w'_2	w'_3
	F_1	F_2	F_3
D_1	30	f_{12}	f_{13}
D_2	47	f_{22}	f_{23}
D_3	39	f_{32}	f_{33}
D_4	50	f_{42}	f_{43}

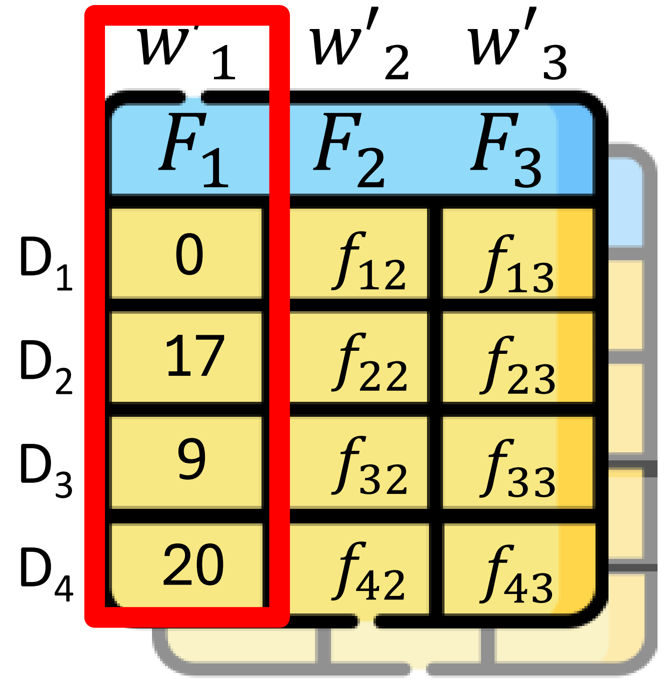
$\Delta_j = f_j - f_{min}$

- F - feature
- D - decompiler
- f - occurrence of the feature
- w - weight of the feature

R2I – R2I Computation

Index computation

e.g. # of tokens



	w'_1	w'_2	w'_3
	F_1	F_2	F_3
D_1	0	f_{12}	f_{13}
D_2	17	f_{22}	f_{23}
D_3	9	f_{32}	f_{33}
D_4	20	f_{42}	f_{43}

$\Delta_j = f_j - f_{min}$

- F - feature
- D - decompiler
- f - occurrence of the feature
- w - weight of the feature

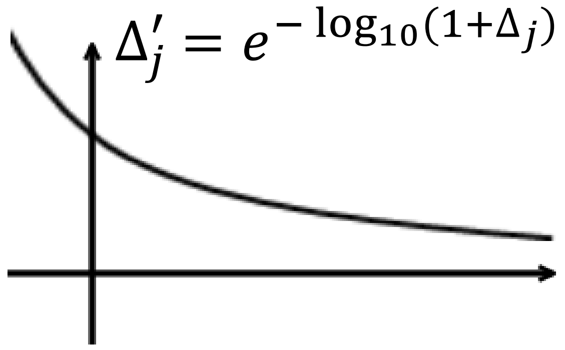
R2I – R2I Computation

▪ Index computation

e.g. # of tokens

	w'_1	w'_2	w'_3
	F_1	F_2	F_3
D_1	0	f_{12}	f_{13}
D_2	17	f_{22}	f_{23}
D_3	9	f_{32}	f_{33}
D_4	20	f_{42}	f_{43}

$\Delta_j = f_j - f_{min}$

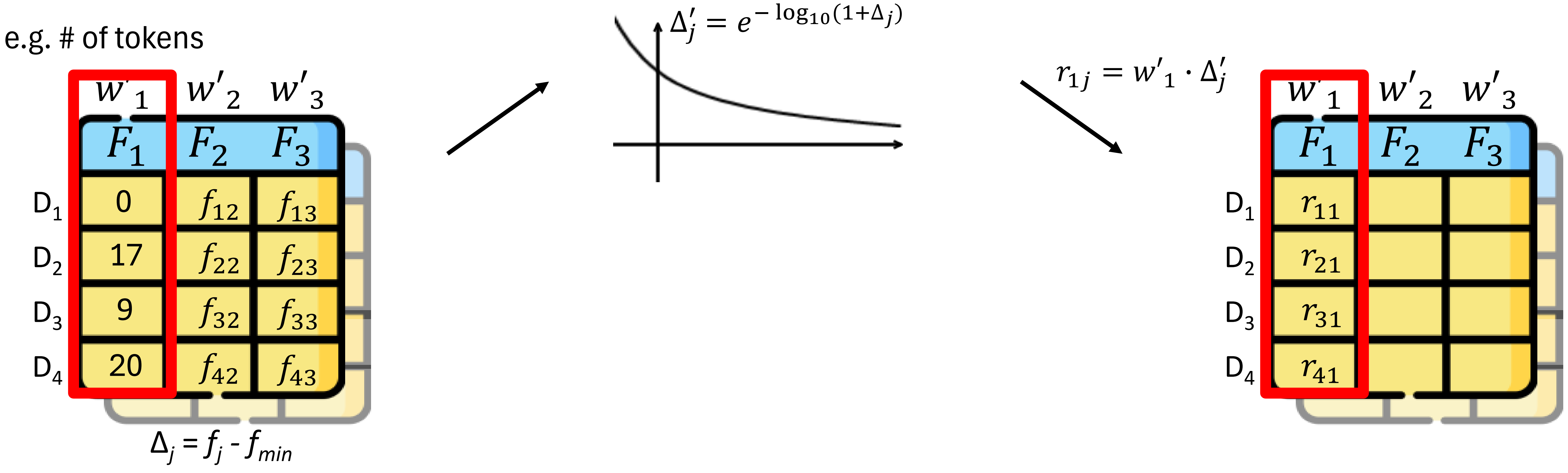


$\Delta'_j = e^{-\log_{10}(1+\Delta_j)}$

- F - feature
- D - decompiler
- f - occurrence of the feature
- w - weight of the feature

R2I – R2I Computation

Index computation



e.g. # of tokens

	w'_1 F_1	w'_2 F_2	w'_3 F_3
D_1	0	f_{12}	f_{13}
D_2	17	f_{22}	f_{23}
D_3	9	f_{32}	f_{33}
D_4	20	f_{42}	f_{43}

$\Delta_j = f_j - f_{min}$

$\Delta'_j = e^{-\log_{10}(1+\Delta_j)}$

$r_{1j} = w'_1 \cdot \Delta'_j$

	w'_1 F_1	w'_2 F_2	w'_3 F_3
D_1	r_{11}		
D_2	r_{21}		
D_3	r_{31}		
D_4	r_{41}		

- F - feature
- D - decompiler
- f - occurrence of the feature
- w - weight of the feature

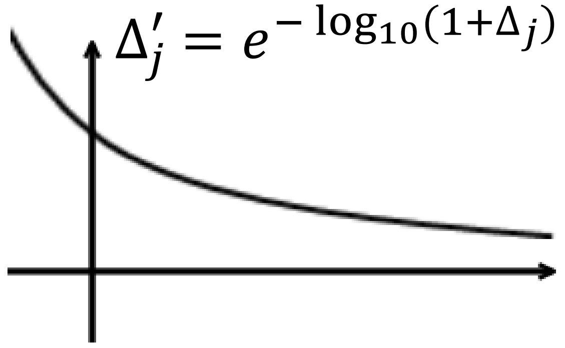
R2I – R2I Computation

▪ Index computation

*(a+8) VS a[1]
e.g. # of arrays

	w'_1	w'_2	w'_3
	F_1	F_2	F_3
D$_1$	0	10	f_{13}
D$_2$	17	2	f_{23}
D$_3$	9	5	f_{33}
D$_4$	20	0	f_{43}

$\Delta_j = f_j - f_{min}$



$\Delta'_j = e^{-\log_{10}(1+\Delta_j)}$

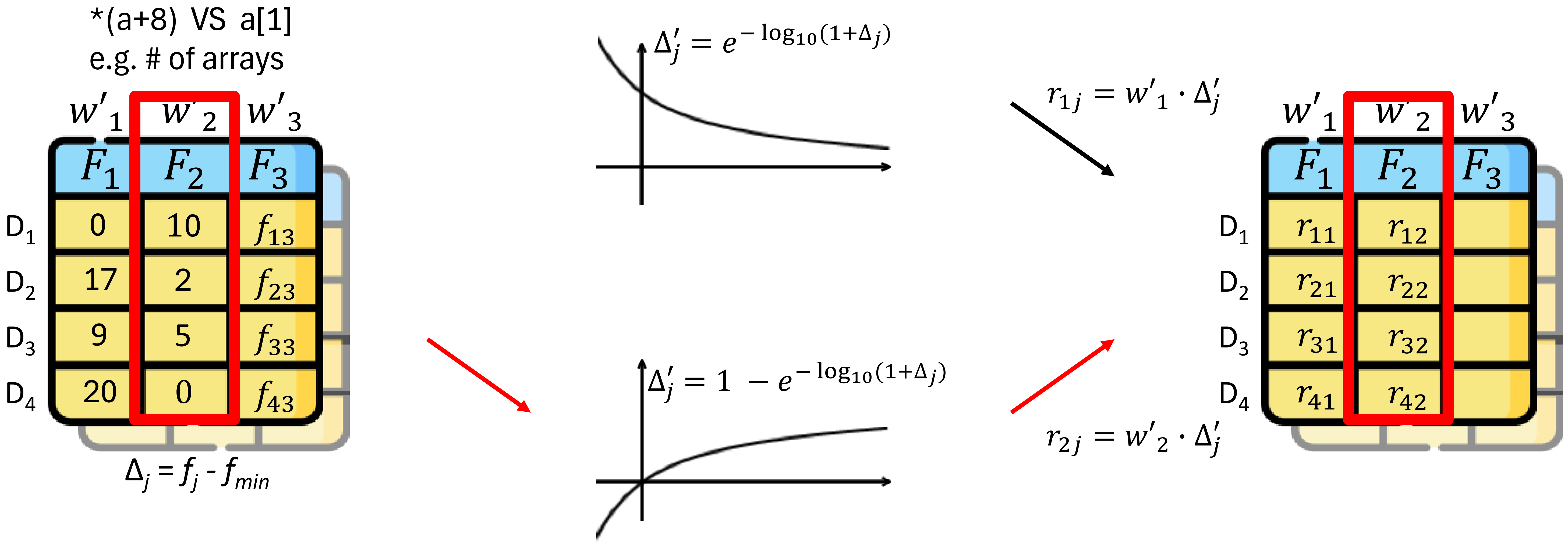
$r_{1j} = w'_1 \cdot \Delta'_j$

	w'_1	w'_2	w'_3
	F_1	F_2	F_3
D$_1$	r_{11}		
D$_2$	r_{21}		
D$_3$	r_{31}		
D$_4$	r_{41}		

- F - feature
- D - decompiler
- f - occurrence of the feature
- w - weight of the feature

R2I – R2I Computation

Index computation



*(a+8) VS a[1]
e.g. # of arrays

	w'_1	w'_2	w'_3
	F_1	F_2	F_3
D$_1$	0	10	f_{13}
D$_2$	17	2	f_{23}
D$_3$	9	5	f_{33}
D$_4$	20	0	f_{43}

$\Delta_j = f_j - f_{min}$

$\Delta'_j = e^{-\log_{10}(1+\Delta_j)}$

$\Delta'_j = 1 - e^{-\log_{10}(1+\Delta_j)}$

$r_{1j} = w'_1 \cdot \Delta'_j$

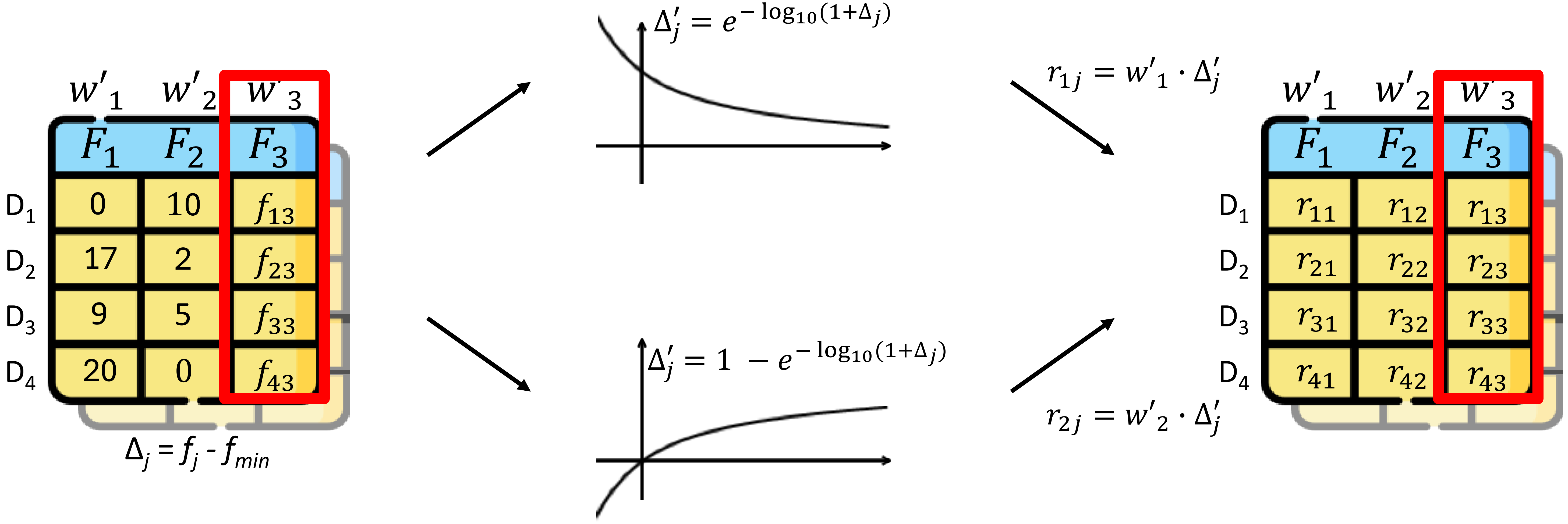
$r_{2j} = w'_2 \cdot \Delta'_j$

	w'_1	w'_2	w'_3
	F_1	F_2	F_3
D$_1$	r_{11}	r_{12}	
D$_2$	r_{21}	r_{22}	
D$_3$	r_{31}	r_{32}	
D$_4$	r_{41}	r_{42}	

- F - feature
- D - decompiler
- f - occurrence of the feature
- w - weight of the feature

R2I – R2I Computation

Index computation



	w'_1 F_1	w'_2 F_2	w'_3 F_3
D_1	0	10	f_{13}
D_2	17	2	f_{23}
D_3	9	5	f_{33}
D_4	20	0	f_{43}

$\Delta_j = f_j - f_{min}$

$\Delta'_j = e^{-\log_{10}(1+\Delta_j)}$

$\Delta'_j = 1 - e^{-\log_{10}(1+\Delta_j)}$

$r_{1j} = w'_1 \cdot \Delta'_j$

$r_{2j} = w'_2 \cdot \Delta'_j$

	w'_1 F_1	w'_2 F_2	w'_3 F_3
D_1	r_{11}	r_{12}	r_{13}
D_2	r_{21}	r_{22}	r_{23}
D_3	r_{31}	r_{32}	r_{33}
D_4	r_{41}	r_{42}	r_{43}

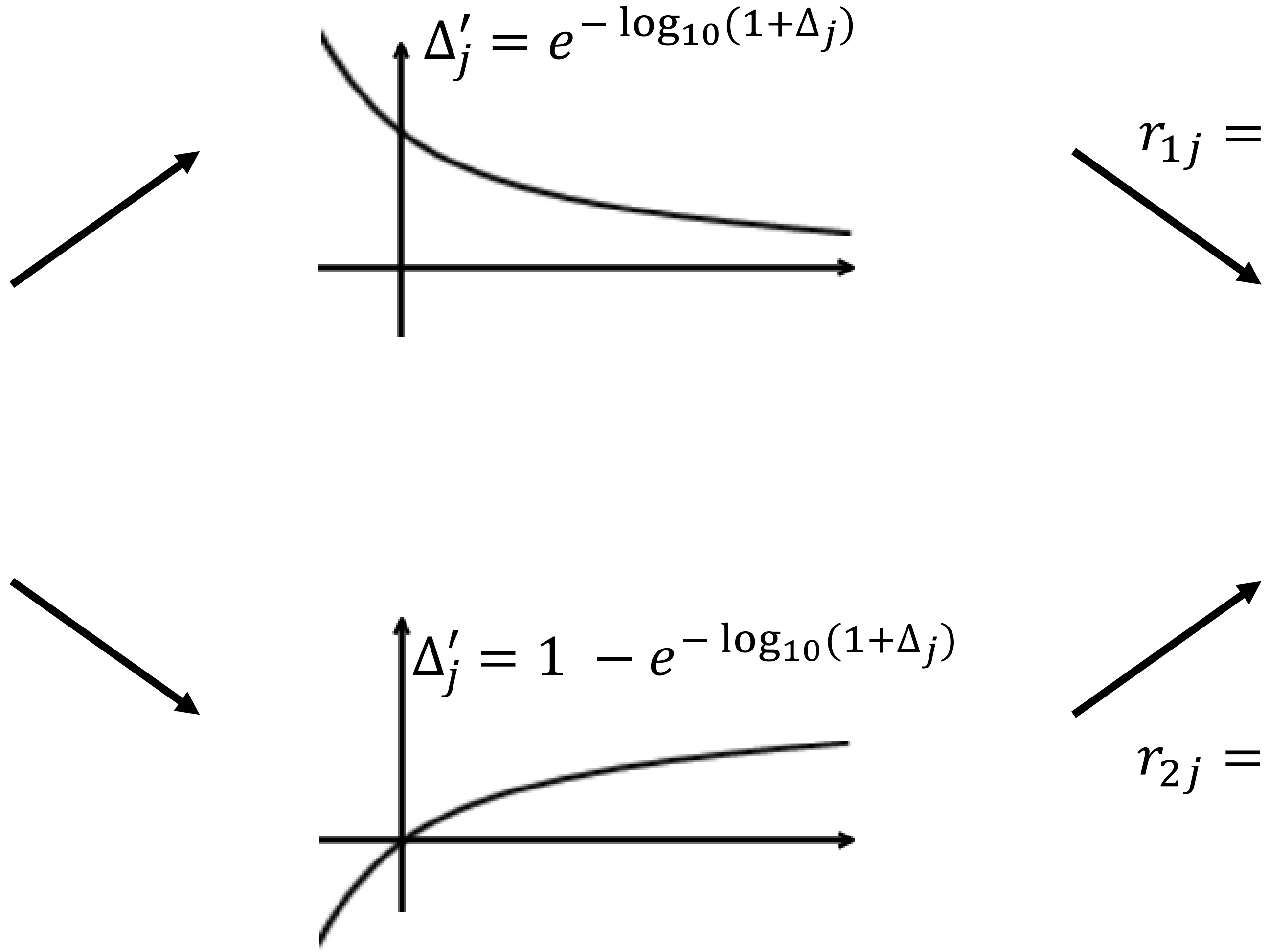
- F - feature
- D - decompiler
- f - occurrence of the feature
- w - weight of the feature

R2I – R2I Computation

▪ Index computation

	w'_1	w'_2	w'_3
	F_1	F_2	F_3
D_1	0	10	f_{13}
D_2	17	2	f_{23}
D_3	9	5	f_{33}
D_4	20	0	f_{43}

$\Delta_j = f_j - f_{min}$



$\Delta'_j = e^{-\log_{10}(1+\Delta_j)}$

$\Delta'_j = 1 - e^{-\log_{10}(1+\Delta_j)}$

$r_{1j} = w'_1 \cdot \Delta'_j$

$r_{2j} = w'_2 \cdot \Delta'_j$

	w'_1	w'_2	w'_3
	F_1	F_2	F_3
D_1	r_{11}	r_{12}	r_{13}
D_2	r_{21}	r_{22}	r_{23}
D_3	r_{31}	r_{32}	r_{33}
D_4	r_{41}	r_{42}	r_{43}

D_i R2I = $\sum_{j=1}^{3} r_{ij}$

- F - feature
- D - decompiler
- f - occurrence of the feature
- w - weight of the feature

Evaluation Setup

Target decompilers

- Hex-Rays, Binary Ninja, Ghidra, Angr, Retdec, Radare2

Dataset

- GNU Coreutils 8.29 & Findutils 4.6.0 compiled with GCC 8.2.0 at the O2 level
    - 103 Coreutils binaries & 4 Findutils binaries
    - 5,305 functions

Evaluation - Practicality

User survey

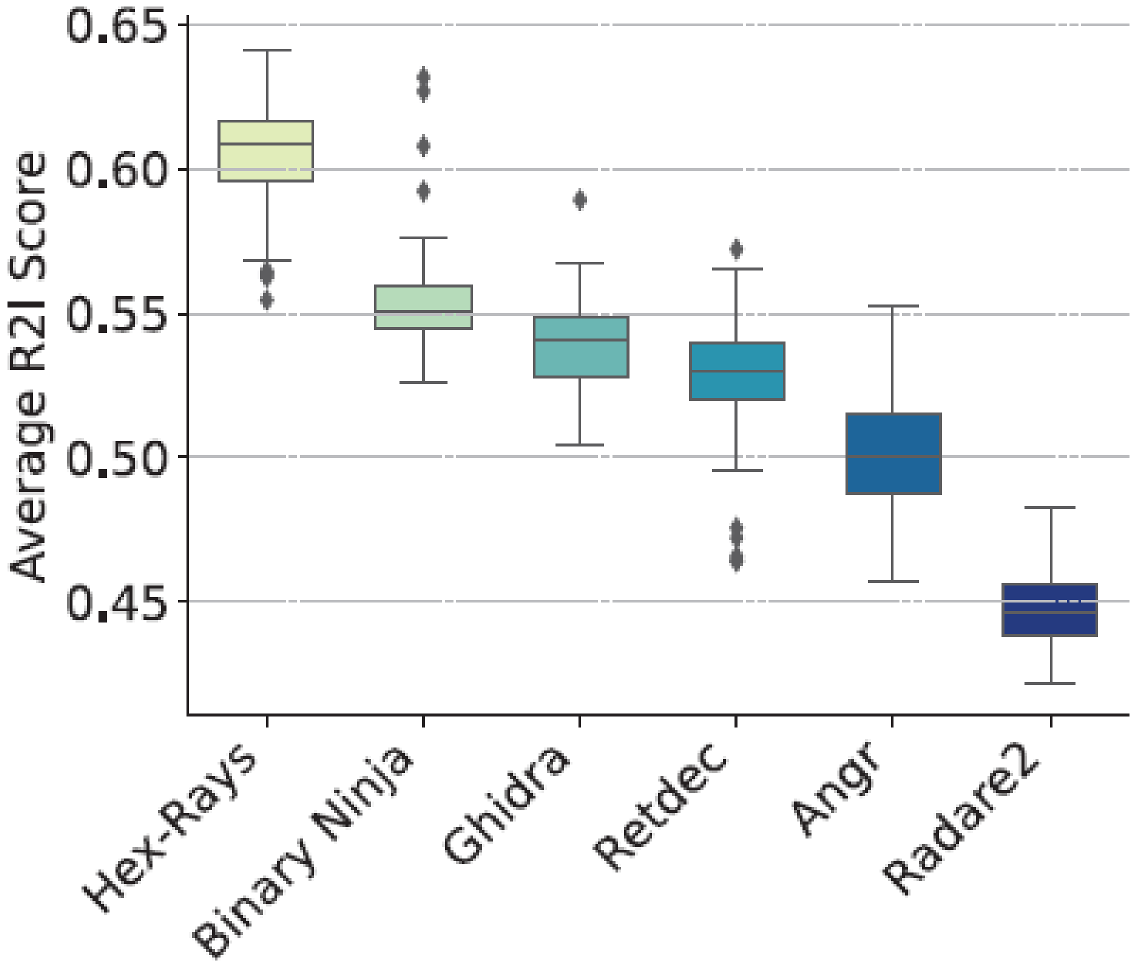
- Purpose
  - Verifying that actual preferences align with R2I indexes
- Participants
  - Recruiting the participants at different skill scales – 22 participants
    - Security engineers, professors engaged in the security field, software engineers, students
  - Reducing a bias towards familiar decompiled code,
    - 45% of participants have less than 6 months of experience with decompilers

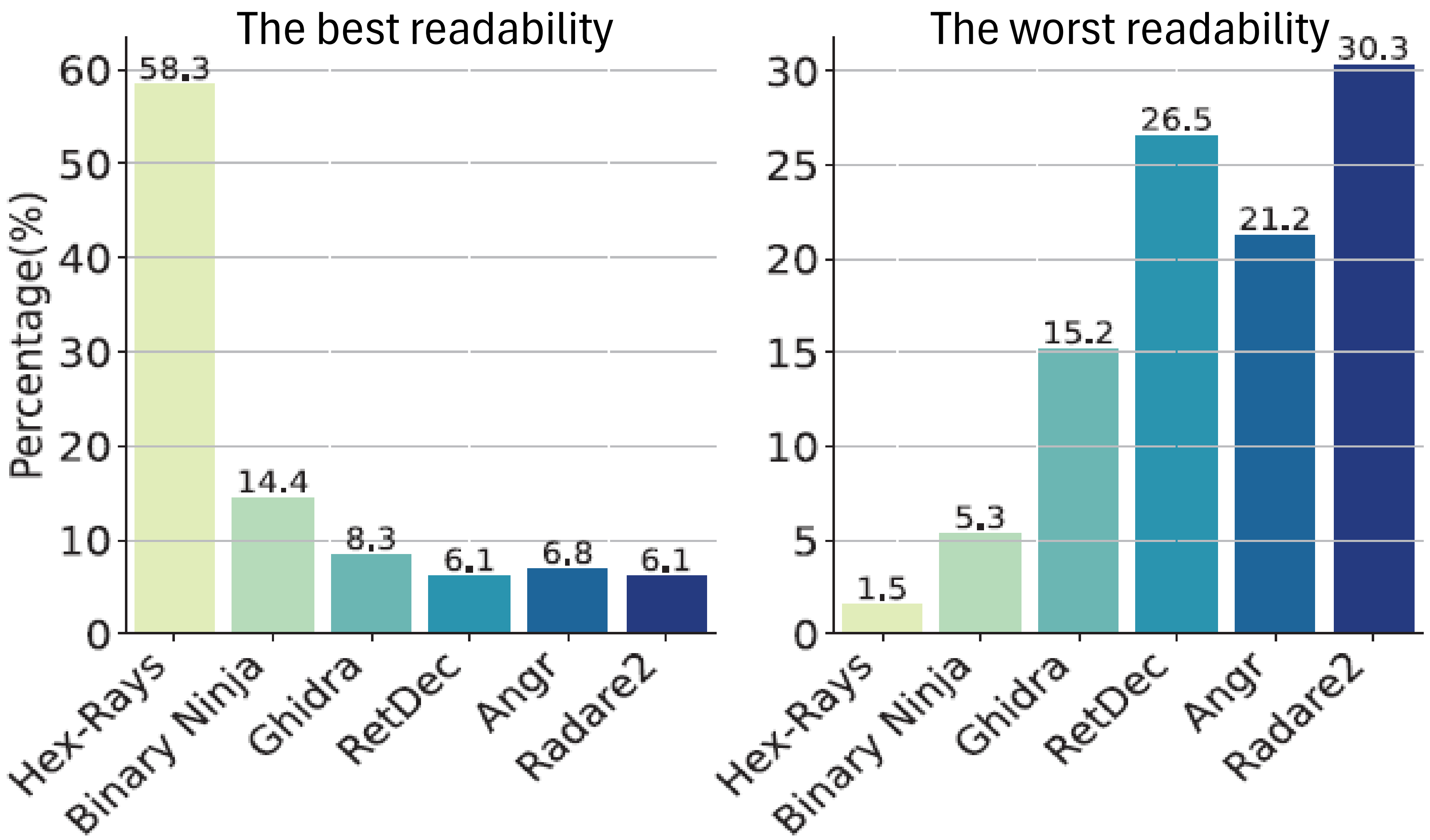
Evaluation - Practicality

▪ Survey design

- Ask to choose the most and least readable decompiled code

▪ Results



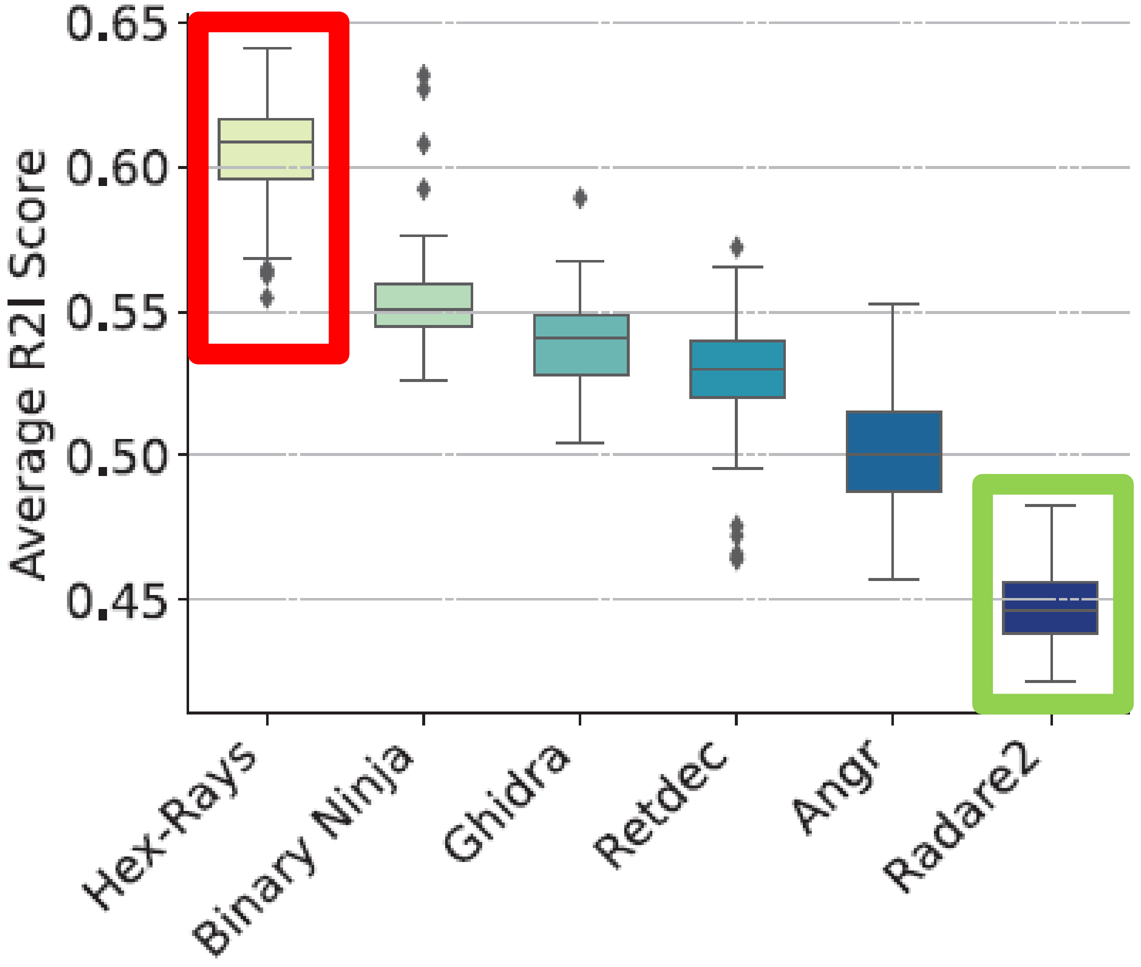


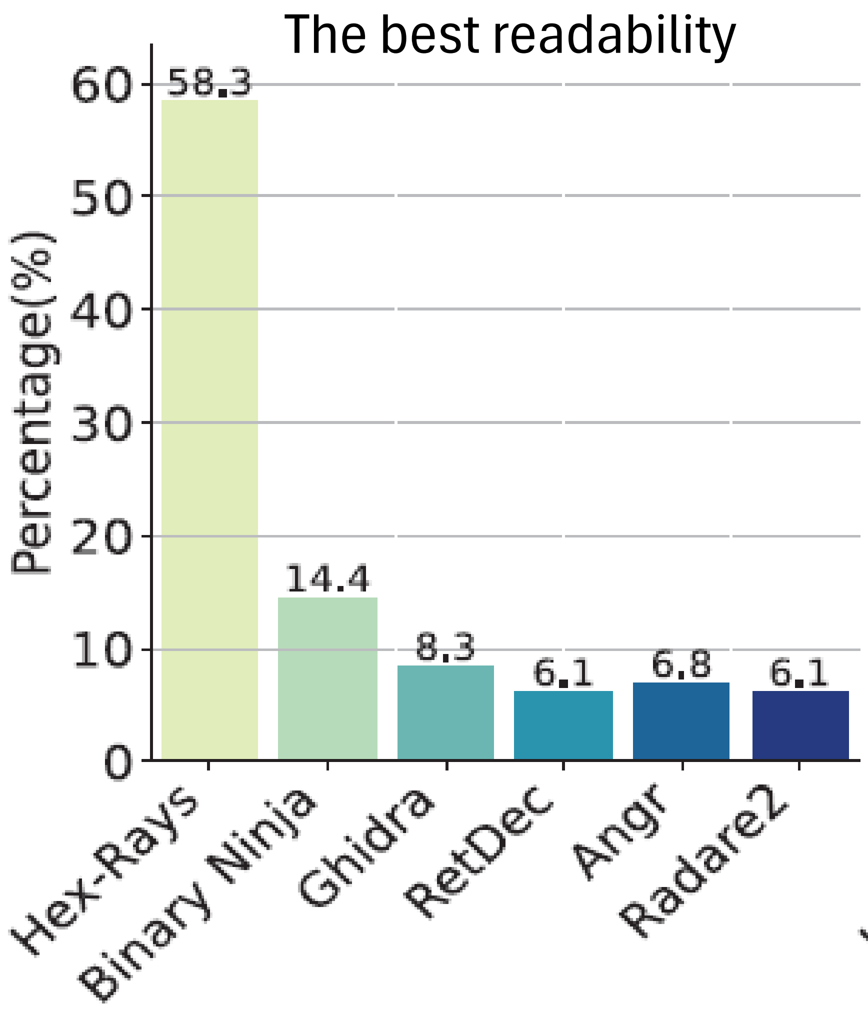
Evaluation - Practicality

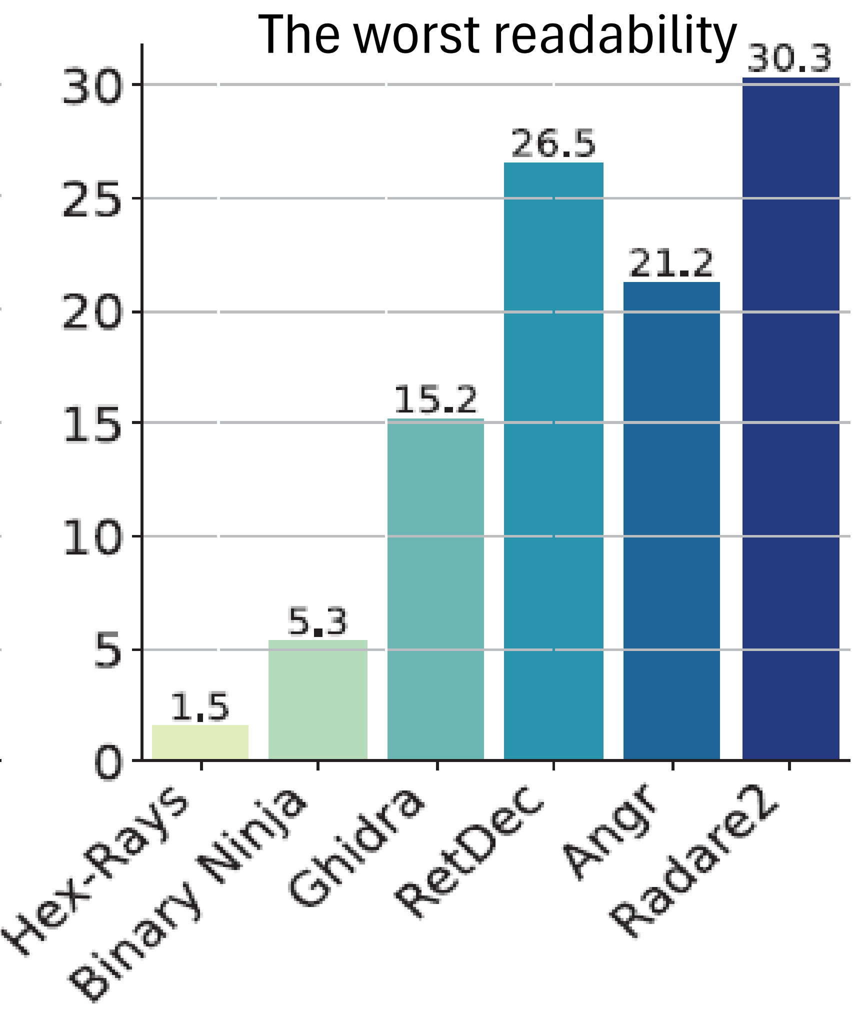
Survey design

- Ask to choose the most and least readable decompiled code

Results





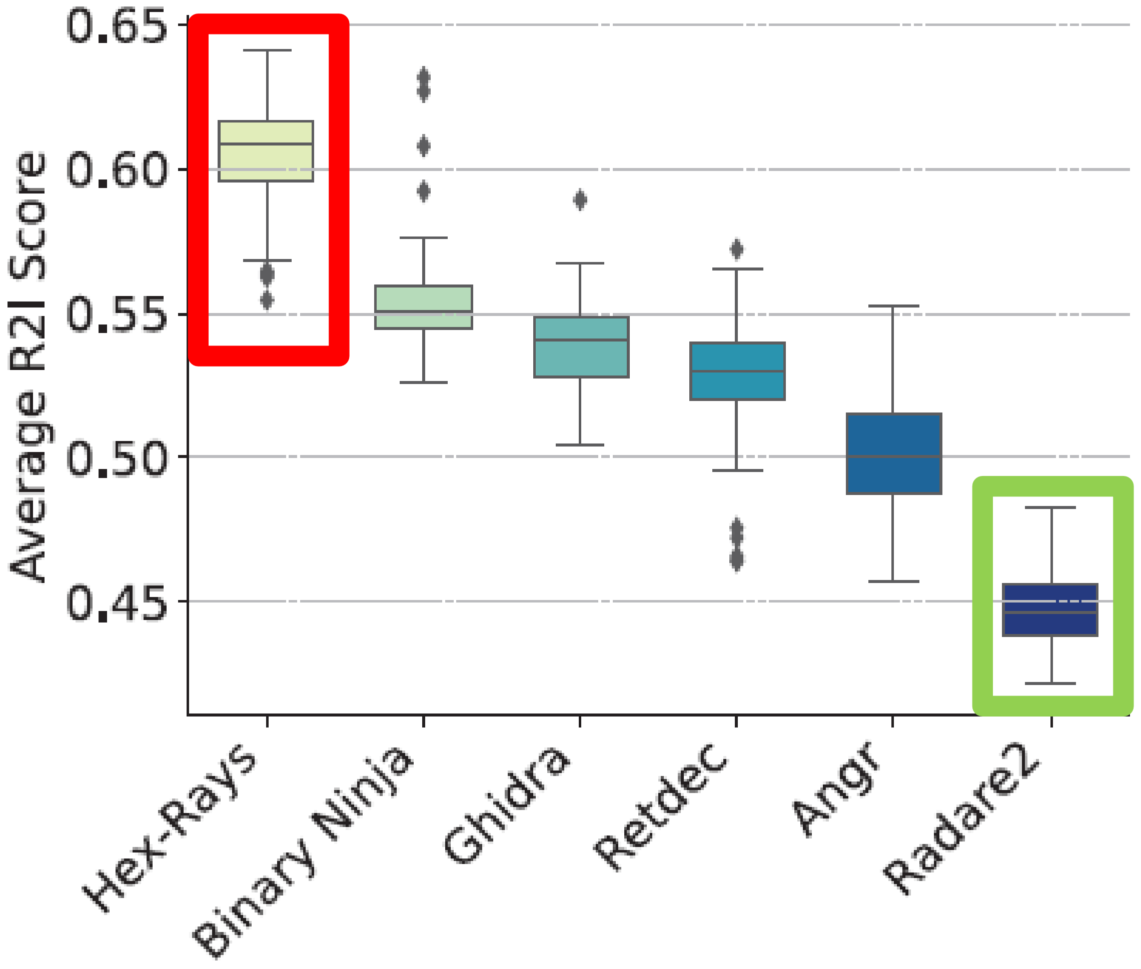


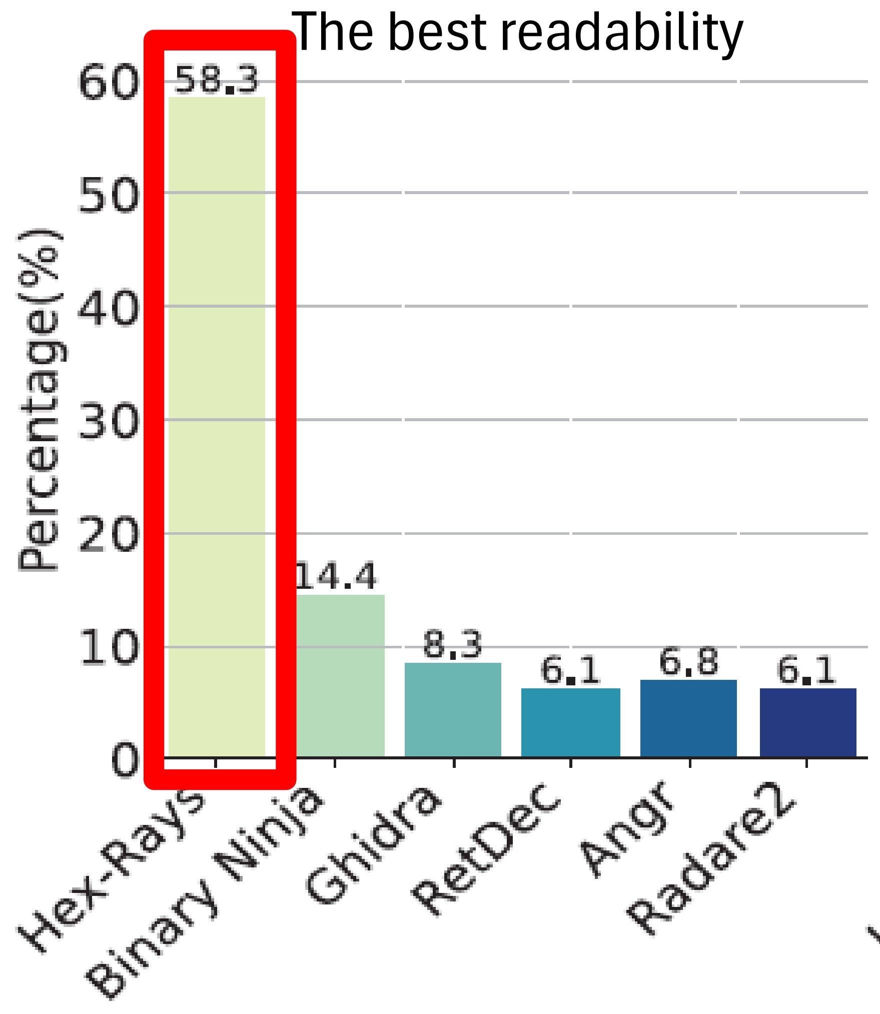
Evaluation - Practicality

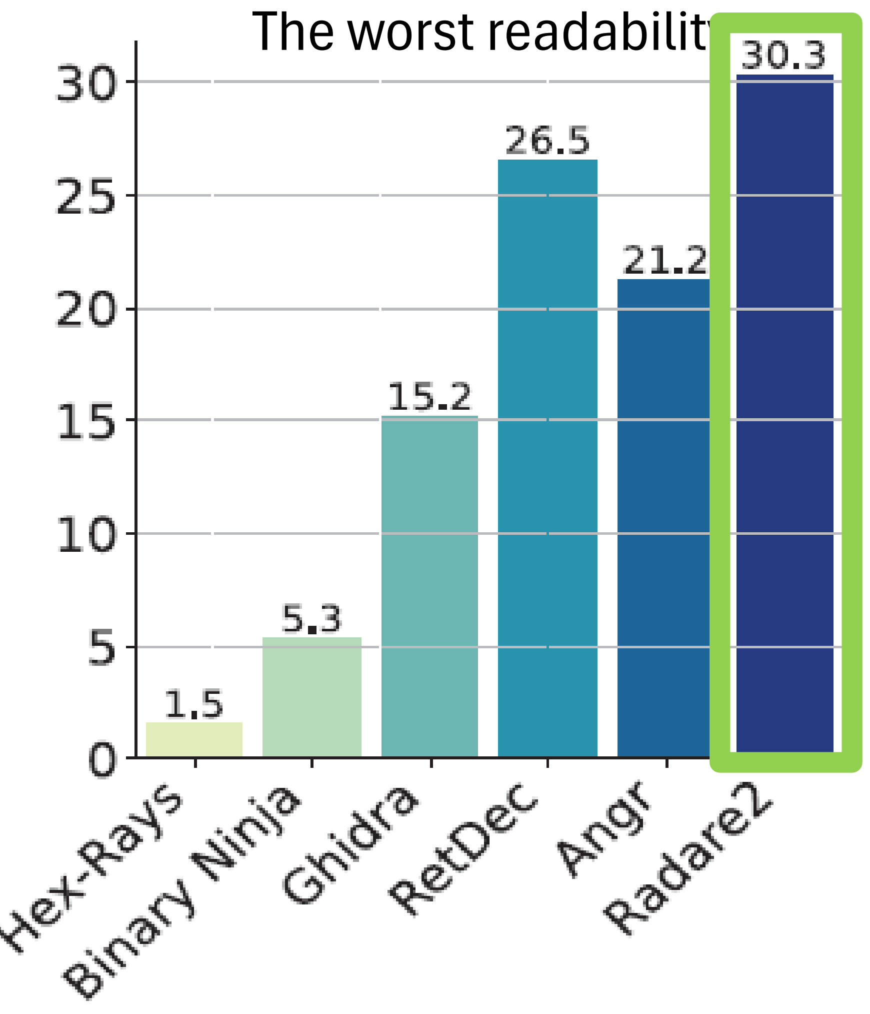
▪ Survey design

- Ask to choose the most and least readable decompiled code

▪ Results



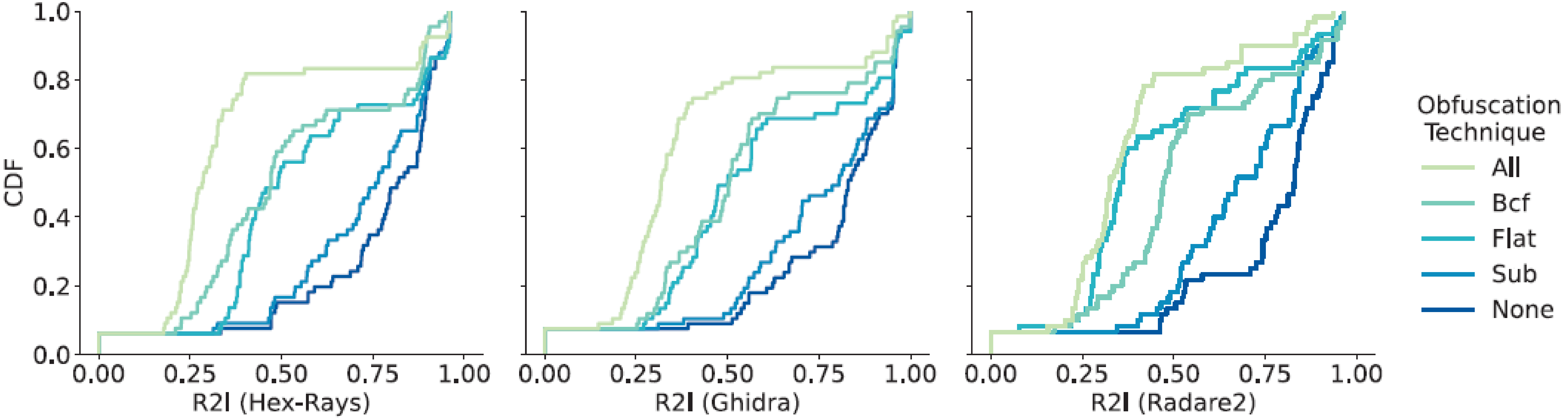




Evaluation - Effectiveness

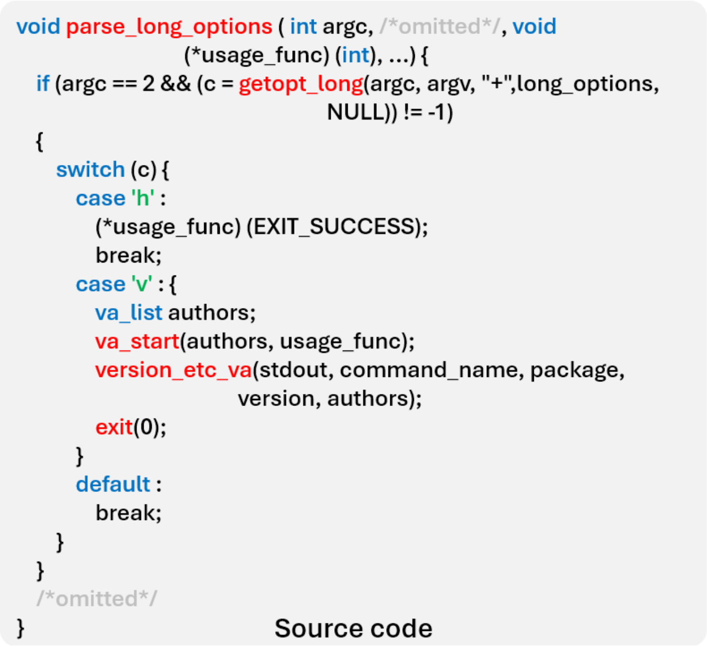
▪ R2I with obfuscated binaries

- Purpose
  - Verifying that non-obfuscated binaries score well
- Results
  - A binary applied all obfuscation techniques has significantly lower R2I scores compared to a non-obfuscated binary



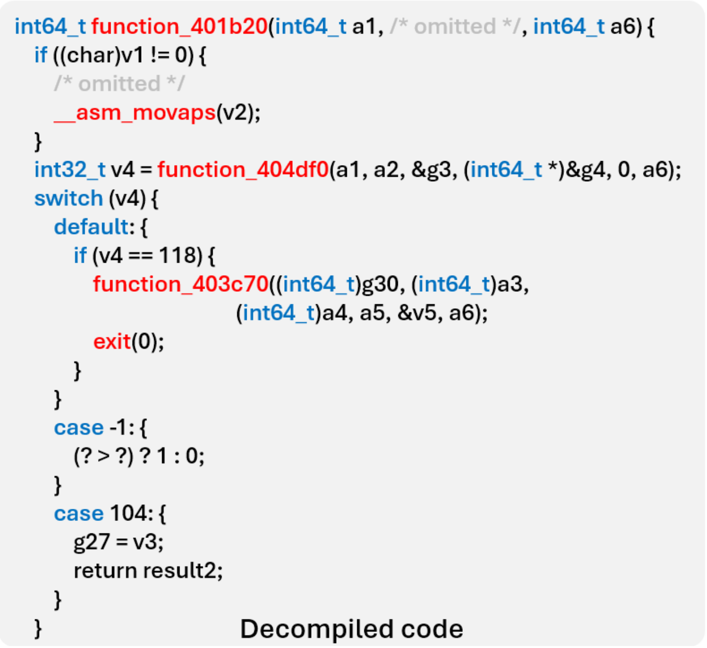
Conclusion

- **Source code vs Decompiled code**



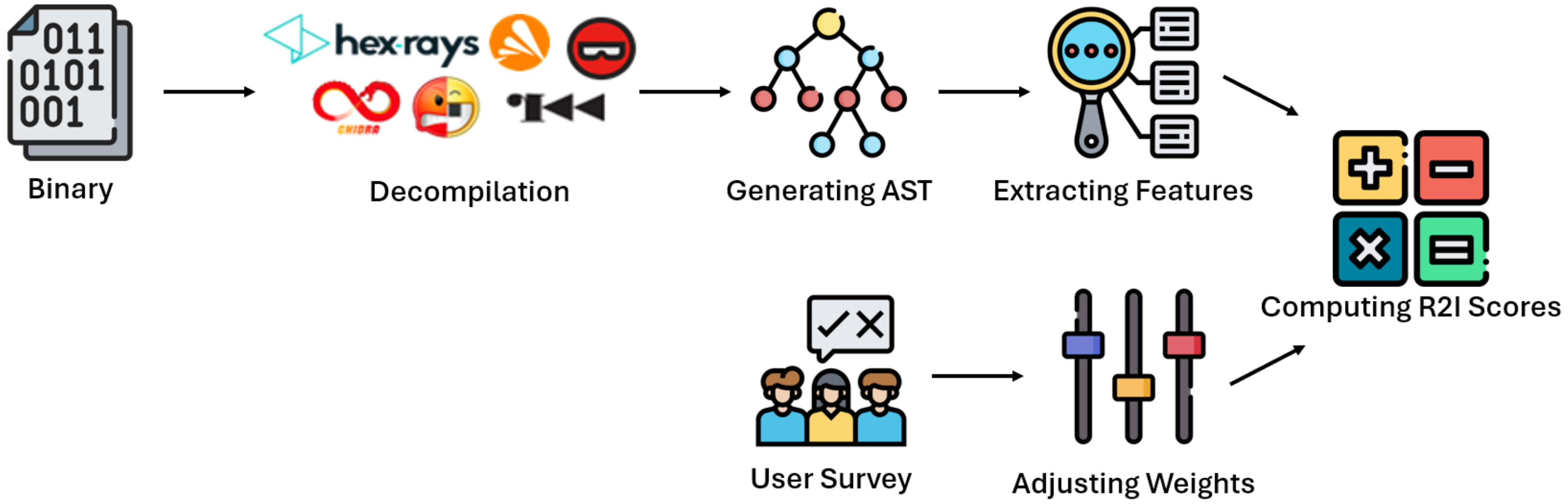
```
void parse_long_options ( int argc, /*omitted*/, void
                (*usage_func) (int), ...) {
  if (argc == 2 && (c = getopt_long(argc, argv, "+",long_options,
                NULL)) != -1)
  {
    switch (c) {
      case 'h' :
        (*usage_func) (EXIT_SUCCESS);
        break;
      case 'v' : {
        va_list authors;
        va_start(authors, usage_func);
        version_etc_va(stdout, command_name, package,
                version, authors);
        exit(0);
      }
      default :
        break;
    }
  }
  /*omitted*/
}
```

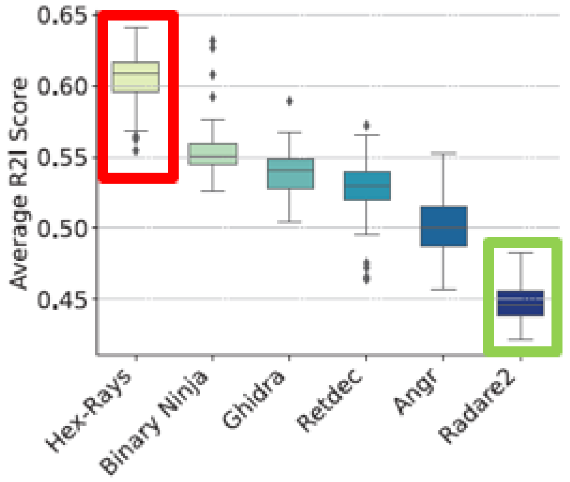
Source code

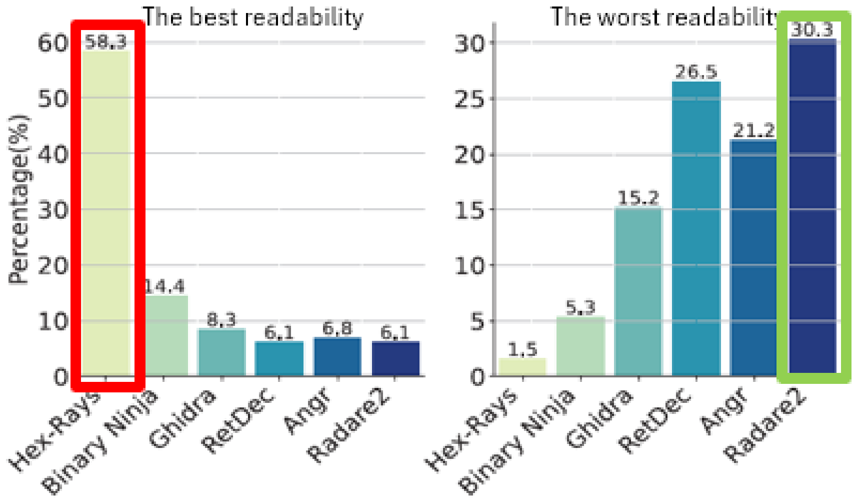


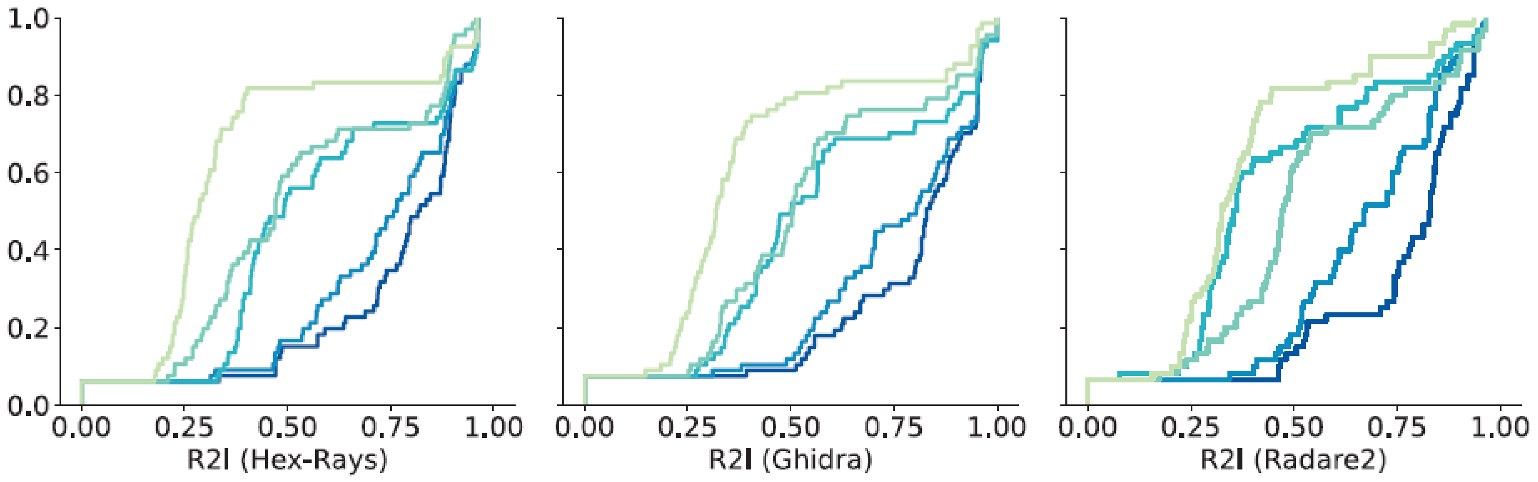
```
int64_t function_401b20(int64_t a1, /* omitted */, int64_t a6) {
  if ((char)v1 != 0) {
    /* omitted */
    __asm_movaps(v2);
  }
  int32_t v4 = function_404df0(a1, a2, &g3, (int64_t *)&g4, 0, a6);
  switch (v4) {
    default: {
      if (v4 == 118) {
        function_403c70((int64_t)g30, (int64_t)a3,
                (int64_t)a4, a5, &v5, a6);
        exit(0);
      }
    }
    case -1: {
      (? > ?) ? 1 : 0;
    }
    case 104: {
      g27 = v3;
      return result2;
    }
}
```

Decompiled code









Thank you