Ahoy SAILR! There is No Need to DREAM of C:

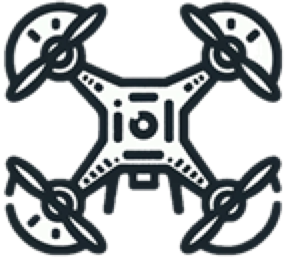
A Compiler-Aware Structuring Algorithm for Binary Decompilation

Zion Leonahenahe Basque, Ati Priya Bajaj, Wil Gibbs, Jude O’Kain, Derron Miao,
Tiffany Bao, Adam Doupé, Yan Shoshitaishvili, Ruoyu Wang

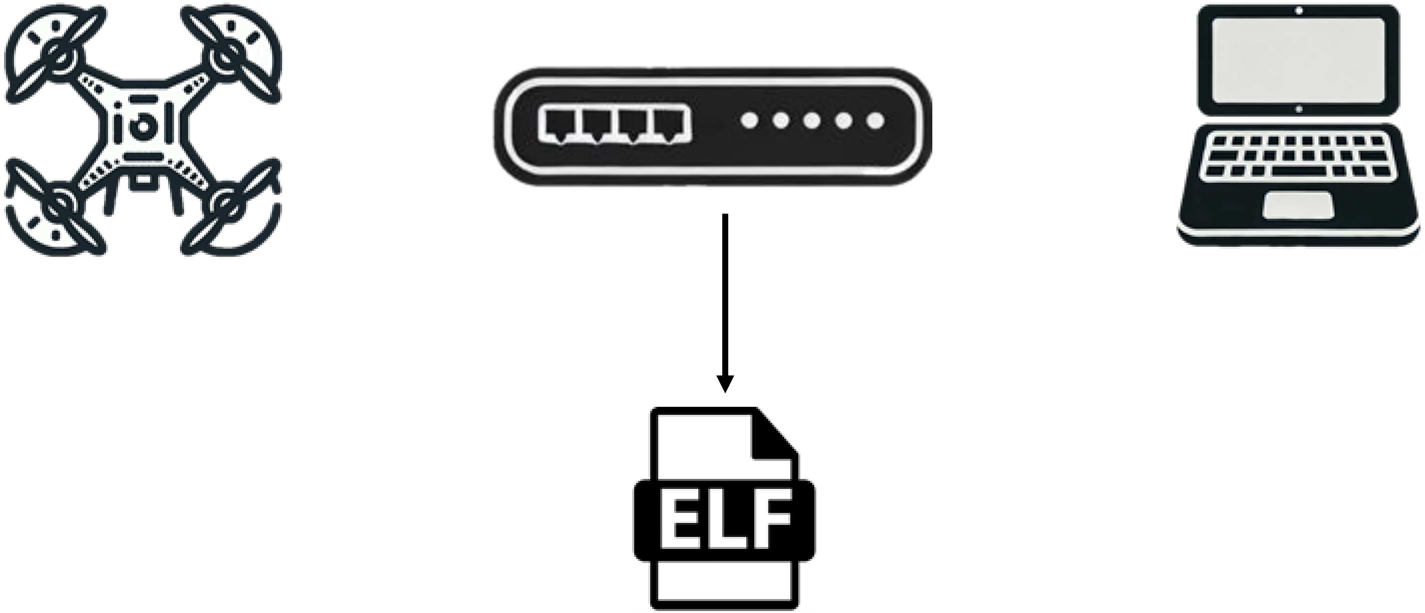
ASU Arizona State University

sefcom
security engineering for future computing

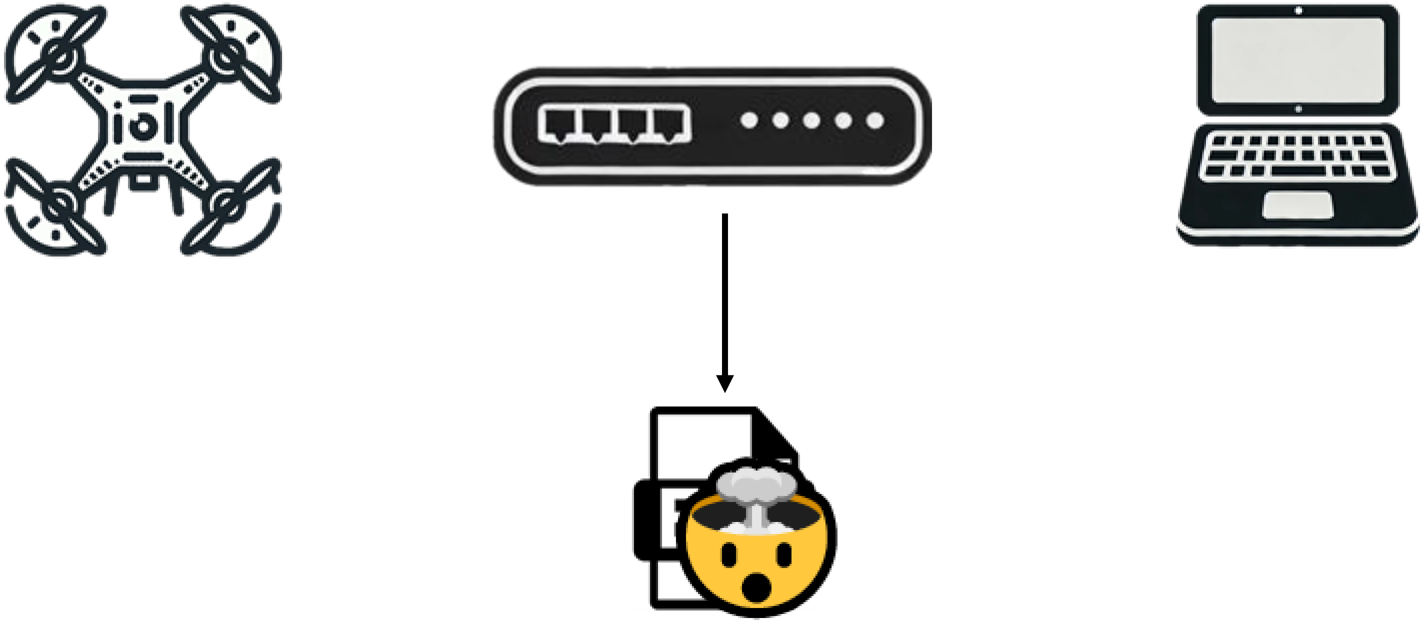
Motivations



Motivations

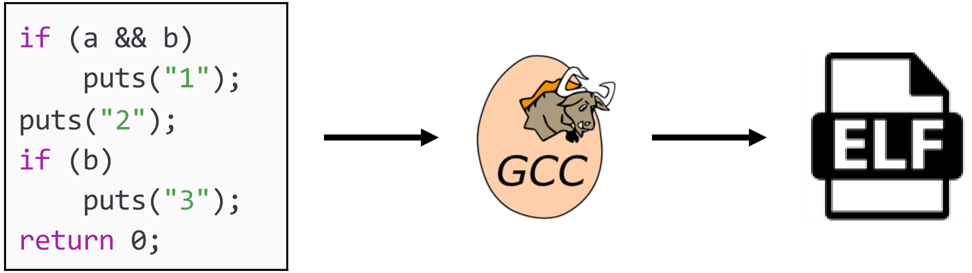


Motivations



Motivations: binaries

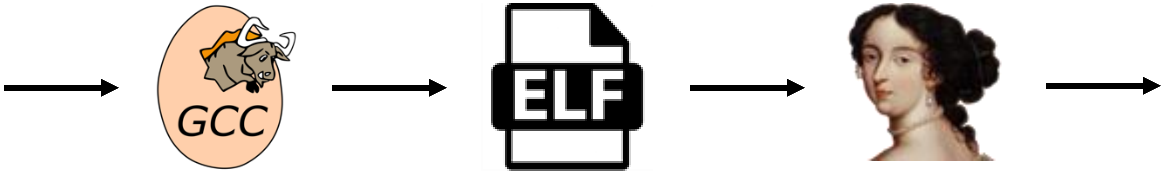
Motivations: binaries



```
if (a && b)
    puts("1");
puts("2");
if (b)
    puts("3");
return 0;
```

Motivations: decompilation

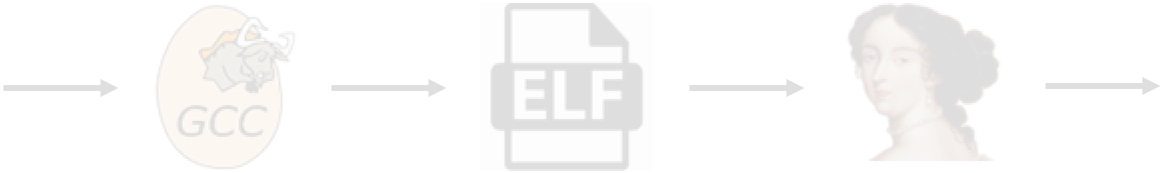
```
if (a && b)
    puts("1");
puts("2");
if (b)
    puts("3");
return 0;
```



```
if (a && b) {
    puts("1");
    puts("2");
    goto B;
}
puts("2");
if (b) {
B:
    puts("3");
}
return 0;
```

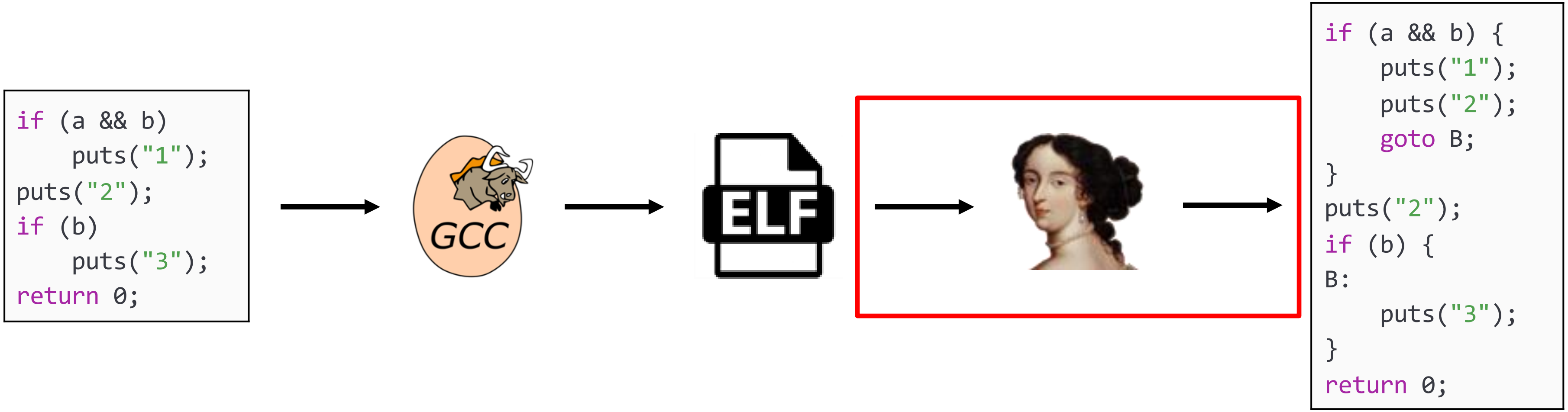
Motivations: decompilation is unlike real source

```
if (a && b)
    puts("1");
puts("2");
if (b)
    puts("3");
return 0;
```



```
if (a && b) {
    puts("1");
    puts("2");
    goto B;
}
puts("2");
if (b) {
B:
    puts("3");
}
return 0;
```

Isolated decompilation research

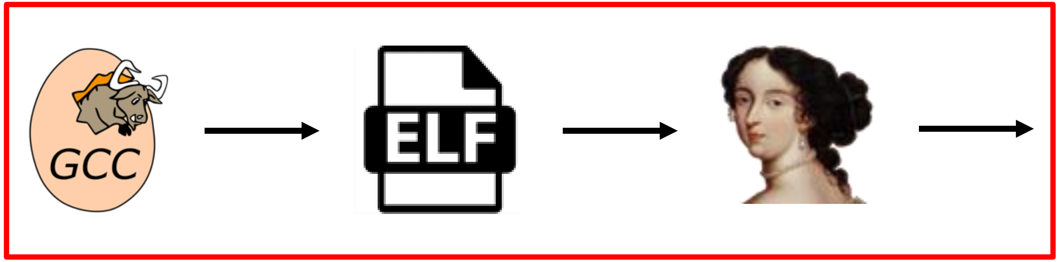


```
if (a && b)
    puts("1");
puts("2");
if (b)
    puts("3");
return 0;
```

```
if (a && b) {
    puts("1");
    puts("2");
    goto B;
}
puts("2");
if (b) {
B:
    puts("3");
}
return 0;
```

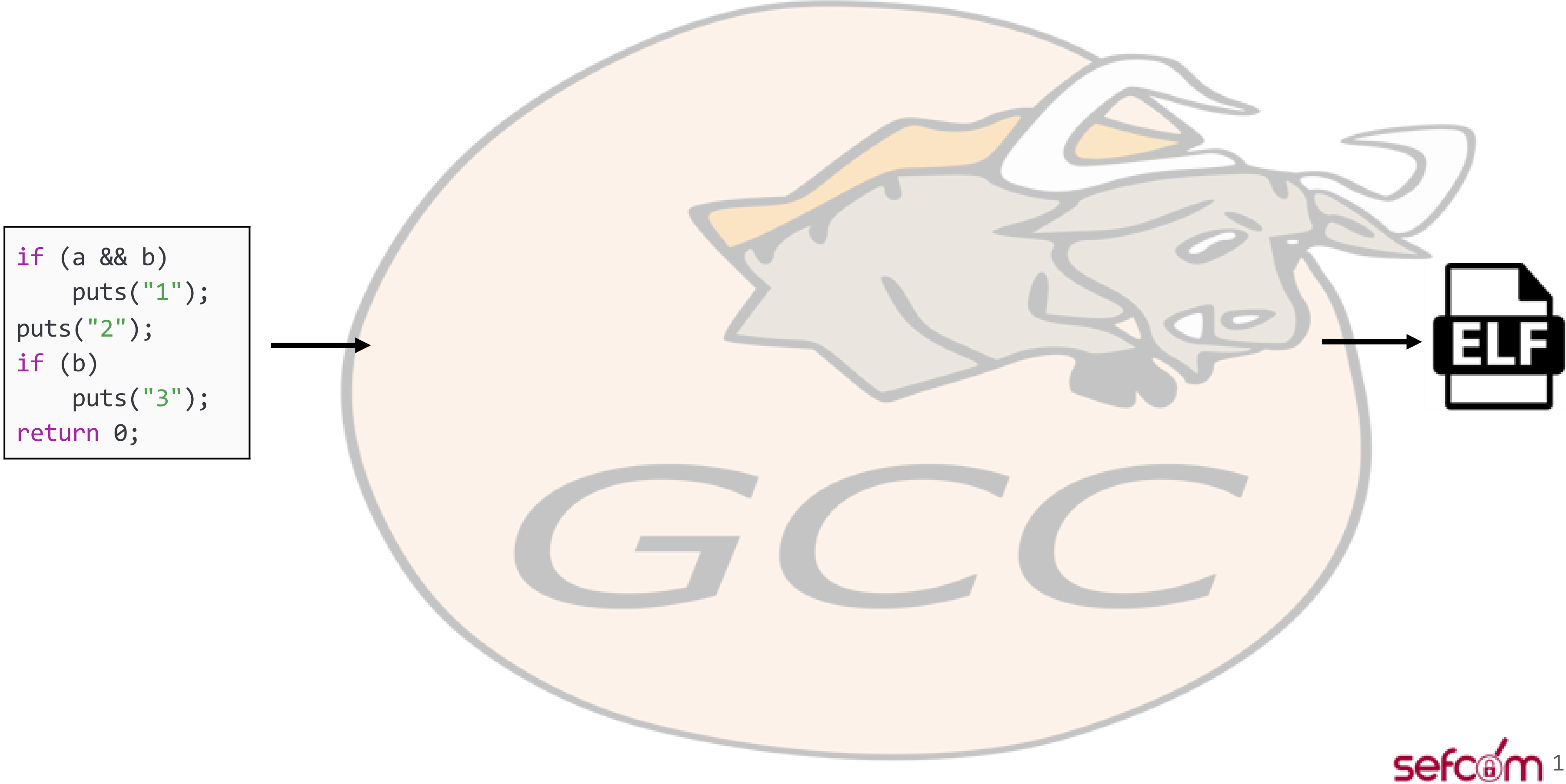
Decompilers should reverse compilation

```
if (a && b)
    puts("1");
puts("2");
if (b)
    puts("3");
return 0;
```



```
if (a && b) {
    puts("1");
    puts("2");
    goto B;
}
puts("2");
if (b) {
B:
    puts("3");
}
return 0;
```

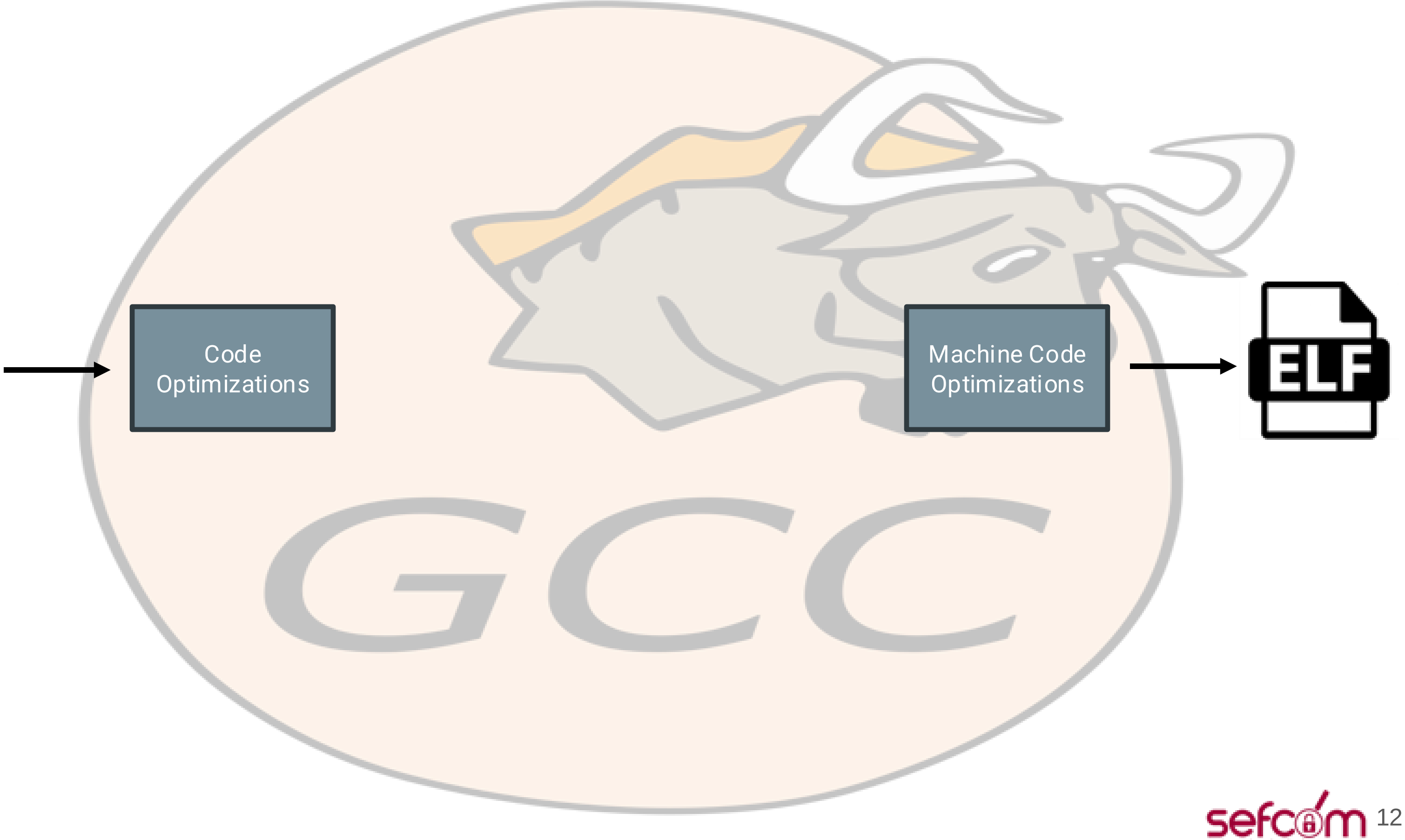
Understanding compilation



```
if (a && b)
    puts("1");
puts("2");
if (b)
    puts("3");
return 0;
```

Understanding compilation

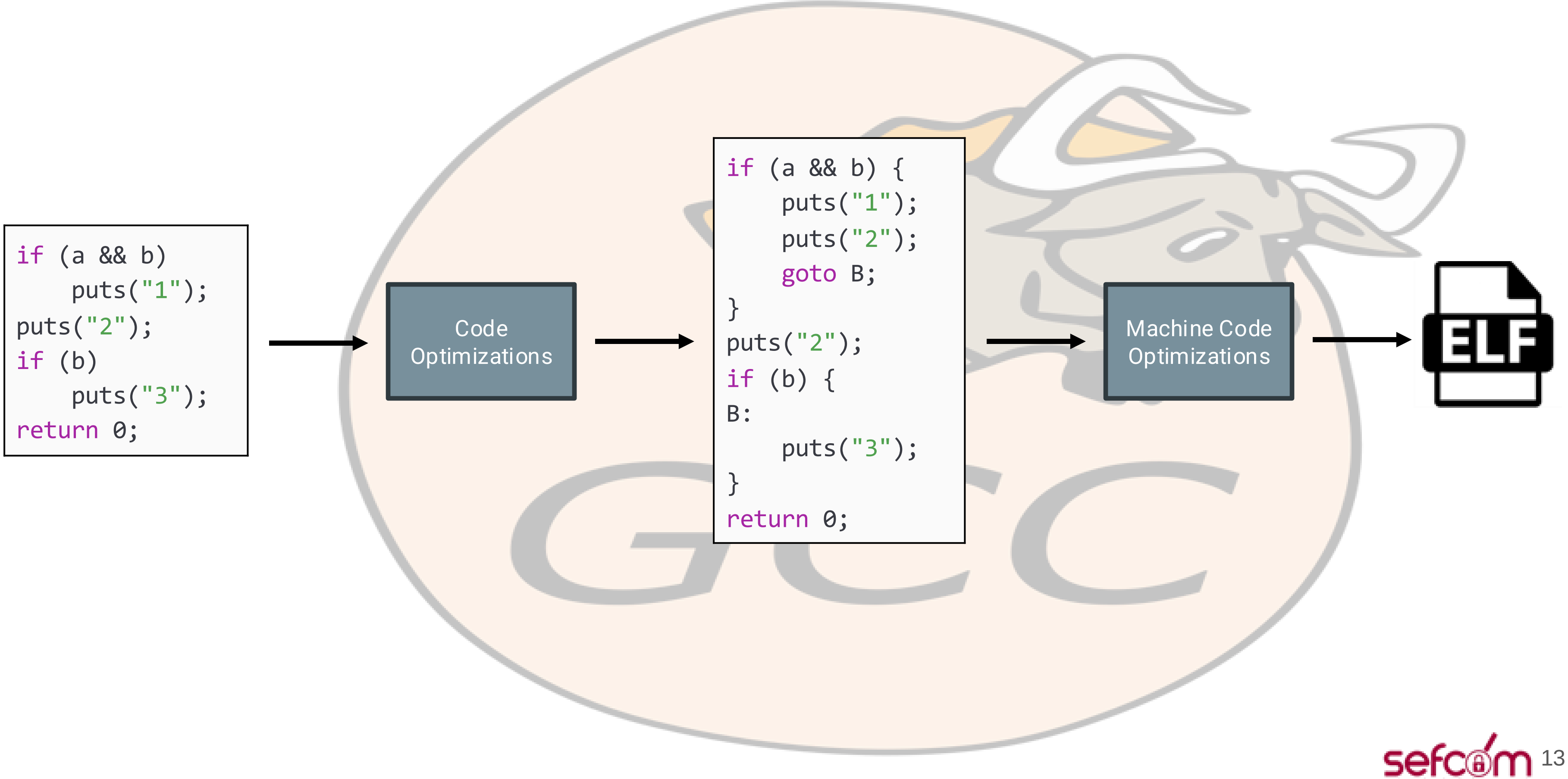
```
if (a && b)
    puts("1");
puts("2");
if (b)
    puts("3");
return 0;
```



Code Optimizations

Machine Code Optimizations

Understanding compilation



```
if (a && b)
    puts("1");
puts("2");
if (b)
    puts("3");
return 0;
```

```
if (a && b) {
    puts("1");
    puts("2");
    goto B;
}
puts("2");
if (b) {
B:
    puts("3");
}
return 0;
```

Modern decompilers

```
if (a && b)
    puts("1");
puts("2");
if (b)
    puts("3");
return 0;
```

```
if (a && b) {
    puts("1");
    puts("2");
    goto B;
}
puts("2");
if (b) {
B:
    puts("3");
}
return 0;
```

Modern Decompilers

Our goal: source-like decompilation

```
if (a && b)
    puts("1");
puts("2");
if (b)
    puts("3");
return 0;
```

The Goal

```
if (a && b) {
    puts("1");
    puts("2");
    goto B;
}
puts("2");
if (b) {
B:
    puts("3");
}
return 0;
```

Modern Decompilers

ELF

Research goals

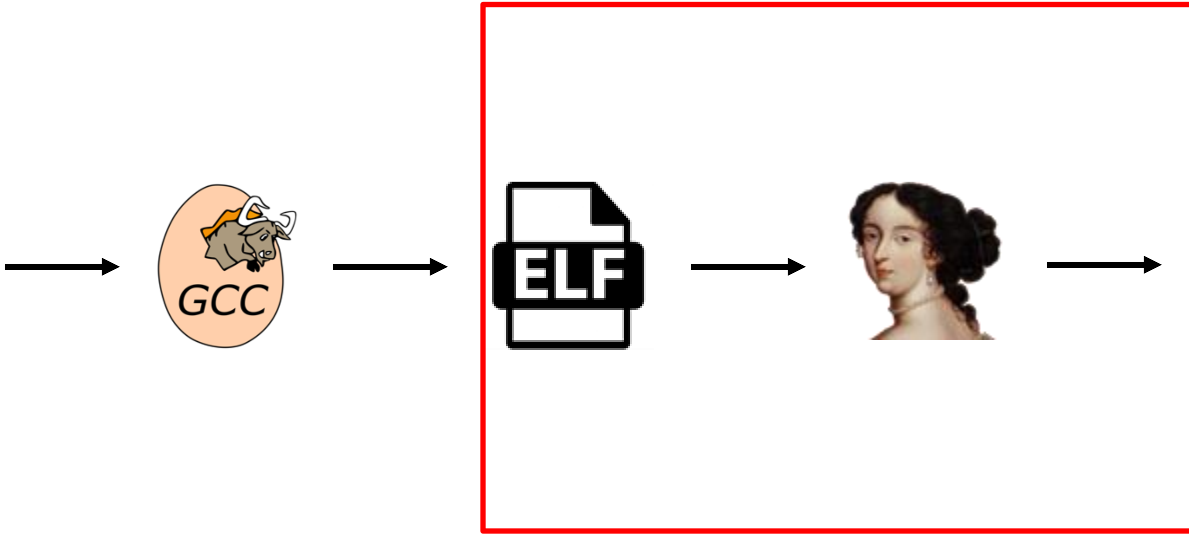
1. **Understand** why decompilation does not look like source

2. **Fix** as many of those problems as we could

3. **Measure** how close decompilers are to recovering perfect source

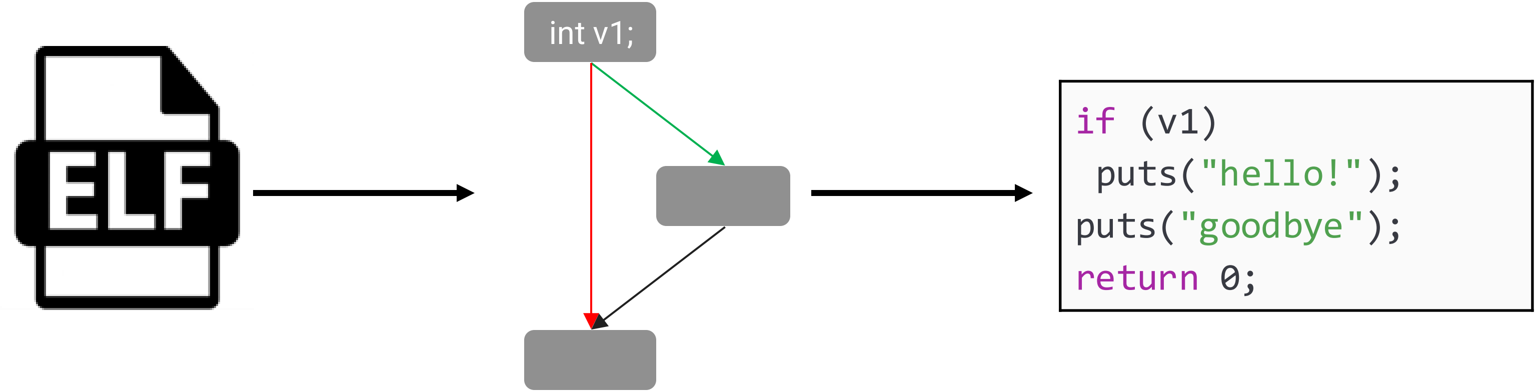
Understanding modern decompilers

```
if (a && b)
    puts("1");
puts("2");
if (b)
    puts("3");
return 0;
```



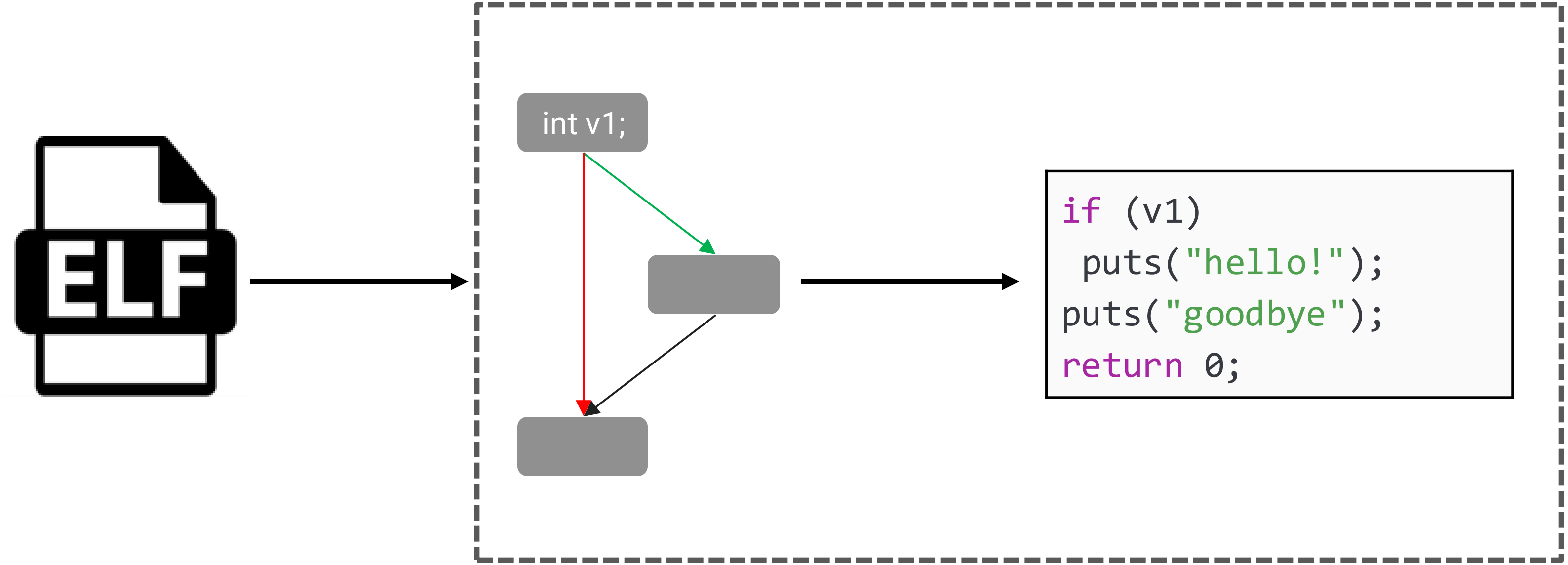
```
if (a && b) {
    puts("1");
    puts("2");
    goto B;
}
puts("2");
if (b) {
B:
    puts("3");
}
return 0;
```

Modern decompilation



```
if (v1)
 puts("hello!");
puts("goodbye");
return 0;
```

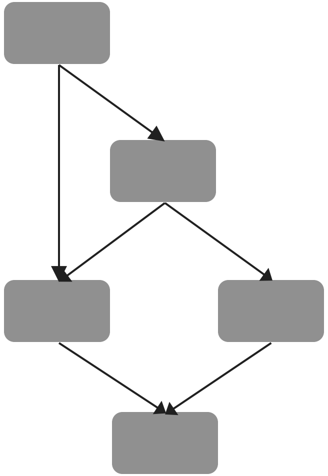
Modern decompilation



```
if (v1)
 puts("hello!");
puts("goodbye");
return 0;
```

Control Flow Structuring

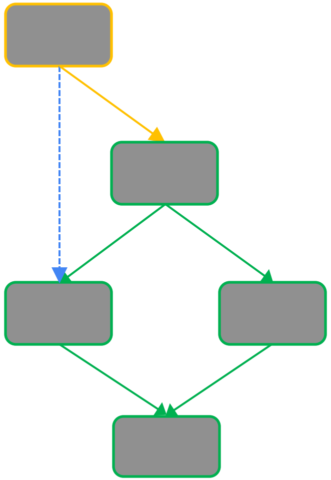
Structuring algorithms



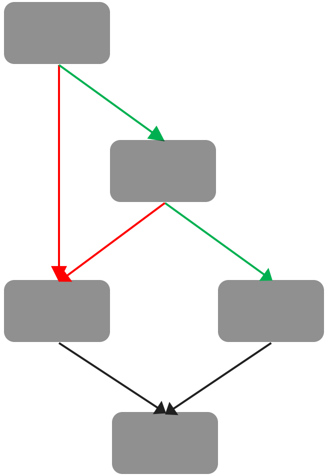
Structuring algorithms: schema-based

Schema-based

```
if (a)
  goto B;
// ...
if (b) {
B:
  // ...
}
else {
  // ...
}
```



Structuring algorithms: gotoless



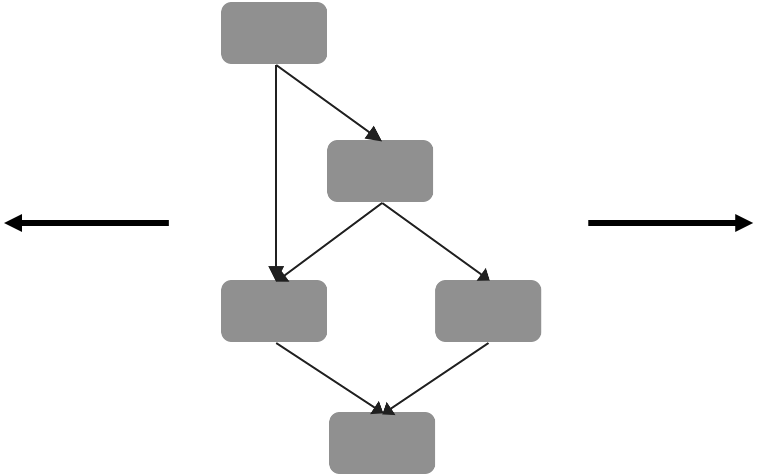
Gotoless

```
if (~a)
  // ...
if (a || b) {
  // ...
}
else {
  // ...
}
```

Structuring algorithms

Schema-based

```
if (a)
  goto B;
// ...
if (b) {
B:
  // ...
}
else {
  // ...
}
```



Gotoless

```
if (~a)
  // ...
if (a || b) {
  // ...
}
else {
  // ...
}
```

Motivations: structuring failures

Source

```
if (a1 && a2)
{
  complete_job();
  if (a3 == EARLY_EXIT)
    goto cleanup;
  next_job();
}
refresh_jobs();
if (a2)
  fast_unlock();
cleanup:
// ...
```

Motivations: structuring failures

Source

```
if (a1 && a2)
{
  complete_job();
  if (a3 == EARLY_EXIT)
    goto cleanup;
  next_job();
}
refresh_jobs();
if (a2)
  fast_unlock();
cleanup:
// ...
```

Hex-Rays [1] (schema)

```
if ( !a1 || !a2 )
{
  refresh_jobs();
  if ( !a2 )
    goto cleanup;
  goto label;
}
complete_job();
if ( EARLY_EXIT != a3 )
{
  next_job();
  refresh_jobs();
label:
  fast_unlock();
}
cleanup:
// ...
```

[1]: The IDA Pro disassembler and debugger. http://www.hex-rays.com/idapro/.

Motivations: structuring failures

Source

```
if (a1 && a2)
{
  complete_job();
  if (a3 == EARLY_EXIT)
    goto cleanup;
  next_job();
}
refresh_jobs();
if (a2)
  fast_unlock();
cleanup:
// ...
```

Hex-Rays [1] (schema)

```
if ( !a1 || !a2 )
{
  refresh_jobs();
  if ( !a2 )
    goto cleanup;
  goto label;
}
complete_job();
if ( EARLY_EXIT != a3 )
{
  next_job();
  refresh_jobs();
label:
  fast_unlock();
}
cleanup:
// ...
```

DREAM [2] (gotoless)

```
if (a1 && a2)
{
  complete_job();
  if (EARLY_EXIT != a3)
  {
    next_job();
    refresh_jobs();
  }
}
if (!a1 || !a2)
  refresh_jobs();
if (a2 && (!a1 || EARLY_EXIT != a3))
  fast_unlock();
// ...
```

[1]: The IDA Pro disassembler and debugger. http://www.hex-rays.com/idapro/.
[2]: Yakdan, Khaled, et al. "No More Gotos: Decompilation Using Pattern-Independent Control-Flow Structuring and Semantic-Preserving Transformations." NDSS. 2015.

Motivations: gotos

Source

```
if (a1 && a2)
{
  complete_job();
  if (a3 == EARLY_EXIT)
    goto cleanup;
  next_job();
}
refresh_jobs();
if (a2)
  fast_unlock();
cleanup:
// ...
```

Hex-Rays [1] (schema)

```
if ( !a1 || !a2 )
{
  refresh_jobs();
  if ( !a2 )
    goto cleanup;
  goto label;
}
complete_job();
if ( EARLY_EXIT != a3 )
{
  next_job();
  refresh_jobs();
label:
  fast_unlock();
}
cleanup:
// ...
```

DREAM [2] (gotoless)

```
if (a1 && a2)
{
  complete_job();
  if (EARLY_EXIT != a3)
  {
    next_job();
    refresh_jobs();
  }
}
if (!a1 || !a2)
  refresh_jobs();
if (a2 && (!a1 || EARLY_EXIT != a3))
  fast_unlock();
// ...
```

[1]: The IDA Pro disassembler and debugger. http://www.hex-rays.com/idapro/.
[2]: Yakdan, Khaled, et al. "No More Gotos: Decompilation Using Pattern-Independent Control-Flow Structuring and Semantic-Preserving Transformations." NDSS. 2015.

Motivations: booleans

Source

```
if (a1 && a2)
{
  complete_job();
  if (a3 == EARLY_EXIT)
    goto cleanup;
  next_job();
}
refresh_jobs();
if (a2)
  fast_unlock();
cleanup:
// ...
```

Hex-Rays [1] (schema)

```
if ( !a1 || !a2 )
{
  refresh_jobs();
  if ( !a2 )
    goto cleanup;
  goto label;
}
complete_job();
if ( EARLY_EXIT != a3 )
{
  next_job();
  refresh_jobs();
label:
  fast_unlock();
}
cleanup:
// ...
```

DREAM [2] (gotoless)

```
if (a1 && a2)
{
  complete_job();
  if (EARLY_EXIT != a3)
  {
    next_job();
    refresh_jobs();
  }
}
if (!a1 || !a2)
  refresh_jobs();
if (a2 && (!a1 || EARLY_EXIT != a3))
  fast_unlock();
// ...
```

[1]: The IDA Pro disassembler and debugger. http://www.hex-rays.com/idapro/.
[2]: Yakdan, Khaled, et al. "No More Gotos: Decompilation Using Pattern-Independent Control-Flow Structuring and Semantic-Preserving Transformations." NDSS. 2015.

Motivations: calls

Source

```
if (a1 && a2)
{
  complete_job();
  if (a3 == EARLY_EXIT)
    goto cleanup;
  next_job();
}
refresh_jobs();
if (a2)
  fast_unlock();
cleanup:
// ...
```

Hex-Rays [1] (schema)

```
if ( !a1 || !a2 )
{
  refresh_jobs();
  if ( !a2 )
    goto cleanup;
  goto label;
}
complete_job();
if ( EARLY_EXIT != a3 )
{
  next_job();
  refresh_jobs();
label:
  fast_unlock();
}
cleanup:
// ...
```

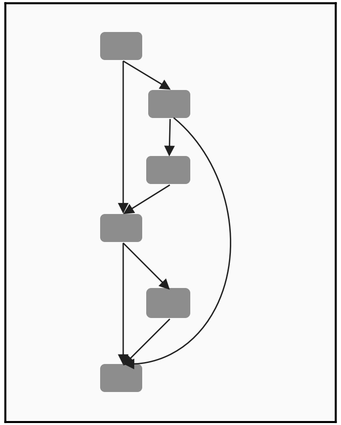
DREAM [2] (gotoless)

```
if (a1 && a2)
{
  complete_job();
  if (EARLY_EXIT != a3)
  {
    next_job();
    refresh_jobs();
  }
}
if (!a1 || !a2)
refresh_jobs();
if (a2 && (!a1 || EARLY_EXIT != a3))
  fast_unlock();
// ...
```

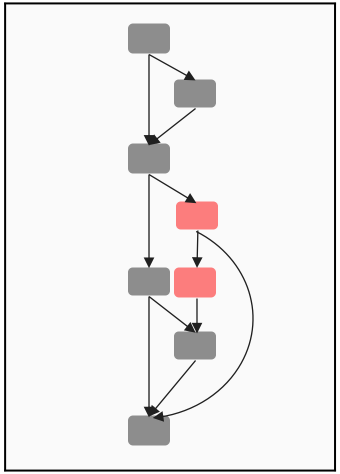
[1]: The IDA Pro disassembler and debugger. http://www.hex-rays.com/idapro/.
[2]: Yakdan, Khaled, et al. "No More Gotos: Decompilation Using Pattern-Independent Control-Flow Structuring and Semantic-Preserving Transformations." NDSS. 2015.

Motivations: code flow

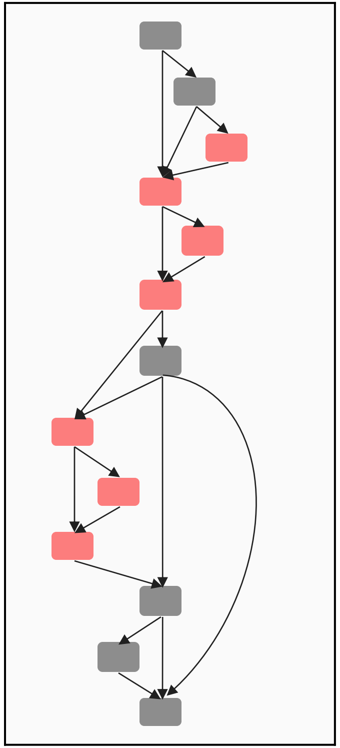
Source



Hex-Rays [1] (schema)



DREAM [2] (gotoless)



[1]: The IDA Pro disassembler and debugger. http://www.hex-rays.com/idapro/.
[2]: Yakdan, Khaled, et al. "No More Gotos: Decompilation Using Pattern-Independent Control-Flow Structuring and Semantic-Preserving Transformations." NDSS. 2015.

Motivations: structuring failures

Source

```
if (a1 && a2)
{
  complete_job();
  if (a3 == EARLY_EXIT)
    goto cleanup;
  next_job();
}
refresh_jobs();
if (a2)
  fast_unlock();
cleanup:
// ...
```

Hex-Rays [1] (schema)

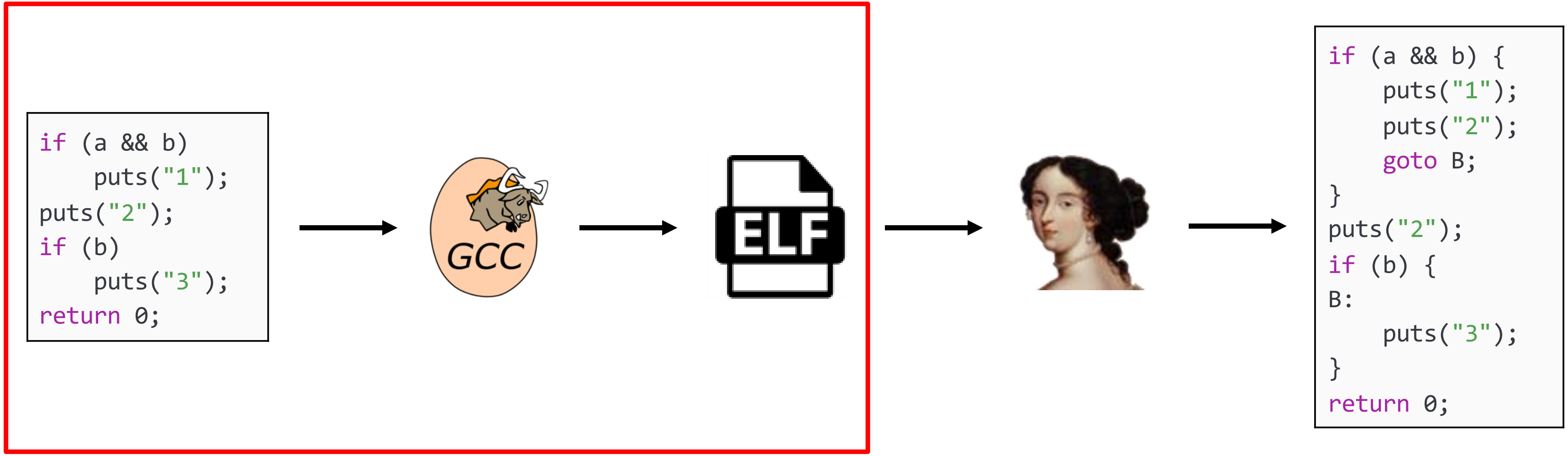
```
if ( !a1 || !a2 )
{
  refresh_jobs();
  if ( !a2 )
    goto cleanup;
  goto label;
}
complete_job();
if ( EARLY_EXIT != a3 )
{
  next_job();
  refresh_jobs();
label:
  fast_unlock();
}
cleanup:
// ...
```

DREAM [2] (gotoless)

```
if (a1 && a2)
{
  complete_job();
  if (EARLY_EXIT != a3)
  {
    next_job();
    refresh_jobs();
  }
}
if (!a1 || !a2)
  refresh_jobs();
if (a2 && (!a1 || EARLY_EXIT != a3))
  fast_unlock();
// ...
```

[1]: The IDA Pro disassembler and debugger. http://www.hex-rays.com/idapro/.
[2]: Yakdan, Khaled, et al. "No More Gotos: Decompilation Using Pattern-Independent Control-Flow Structuring and Semantic-Preserving Transformations." NDSS. 2015.

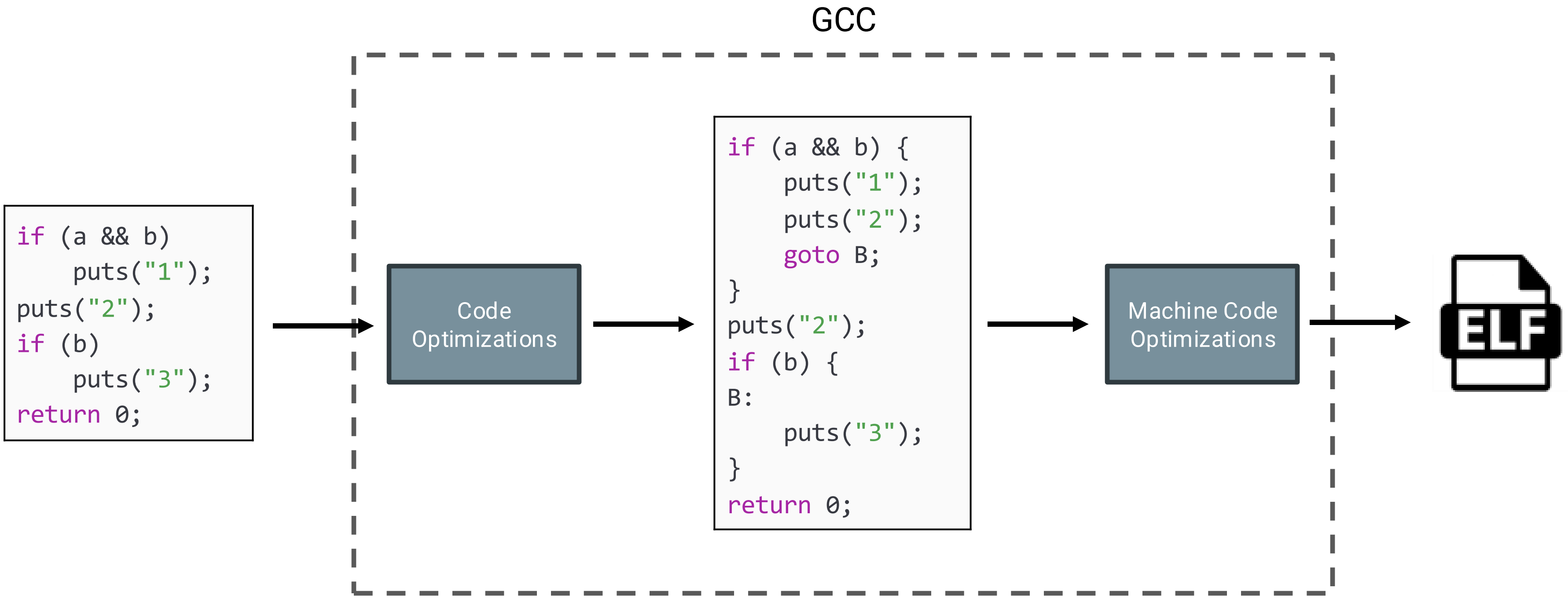
Understanding modern compilers



```
if (a && b)
    puts("1");
puts("2");
if (b)
    puts("3");
return 0;
```

```
if (a && b) {
    puts("1");
    puts("2");
    goto B;
}
puts("2");
if (b) {
B:
    puts("3");
}
return 0;
```

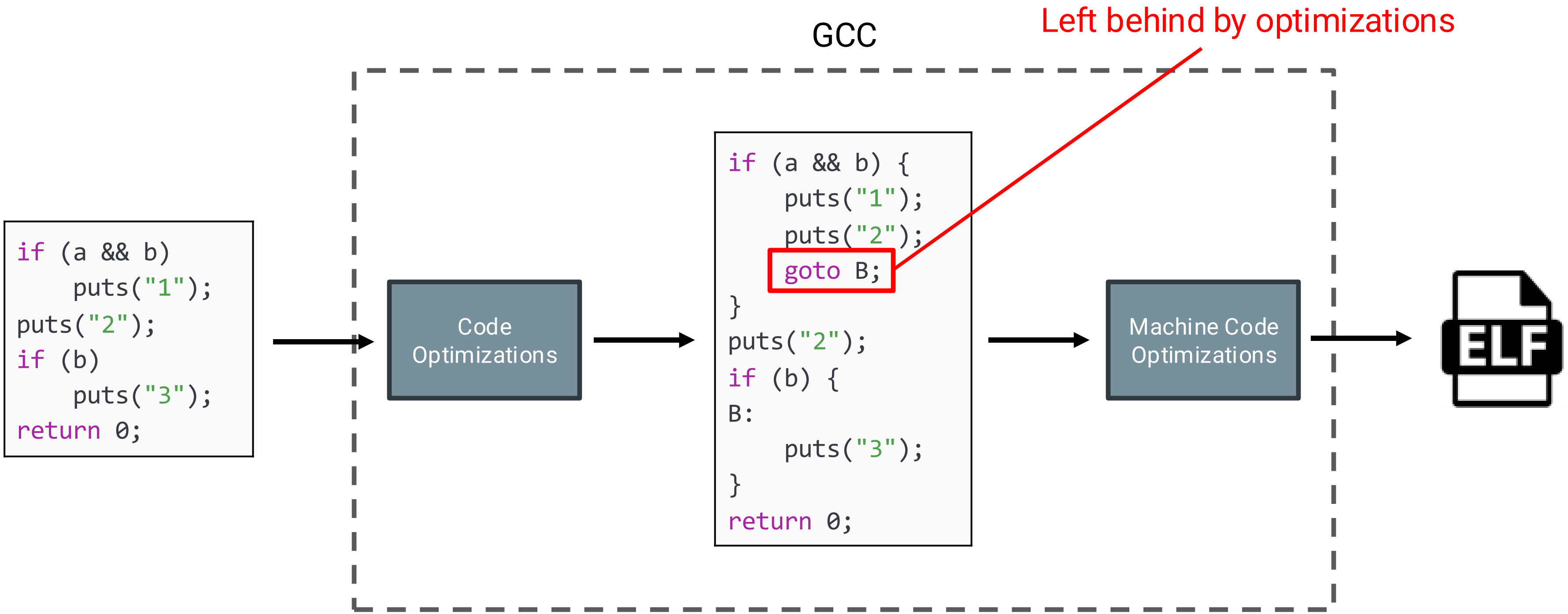
Understanding modern compilers



```
if (a && b)
    puts("1");
puts("2");
if (b)
    puts("3");
return 0;
```

```
if (a && b) {
    puts("1");
    puts("2");
    goto B;
}
puts("2");
if (b) {
B:
    puts("3");
}
return 0;
```

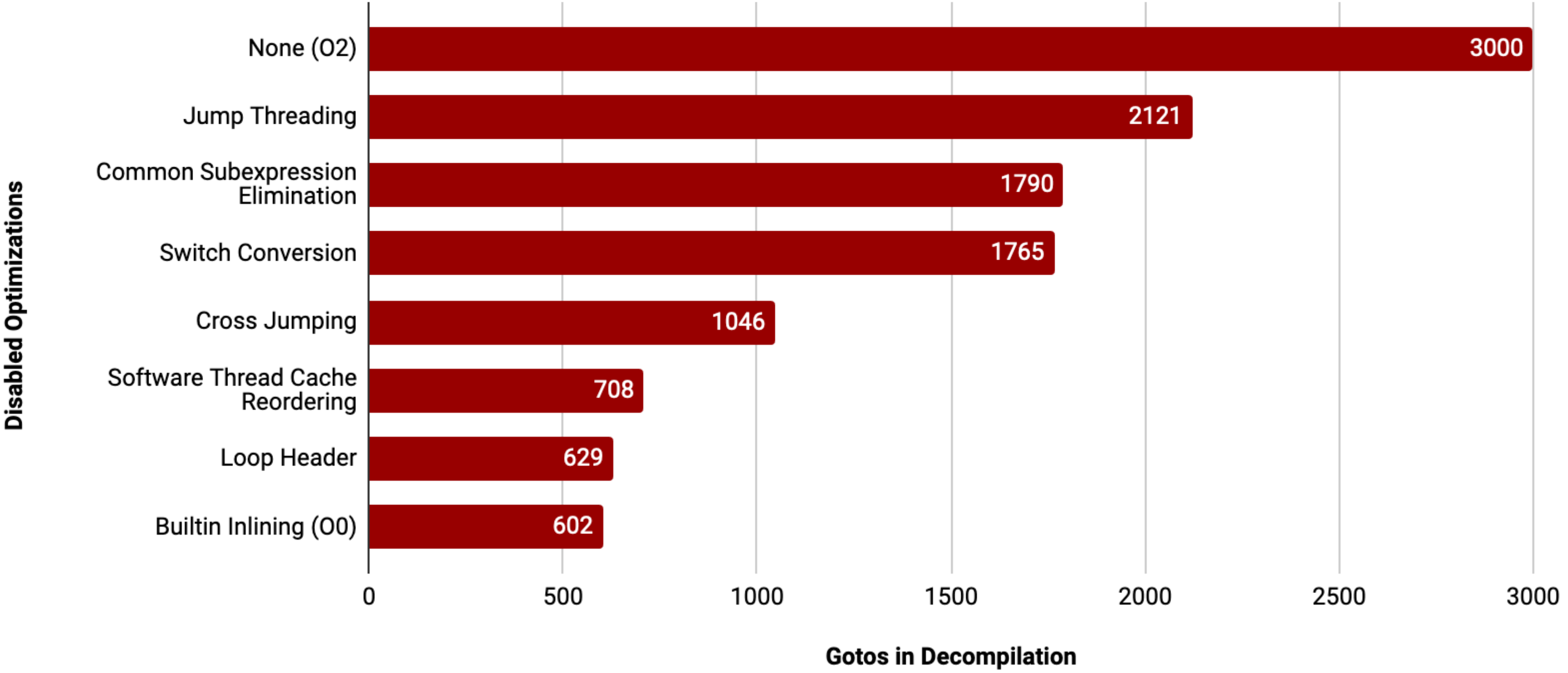
Understanding modern compilers



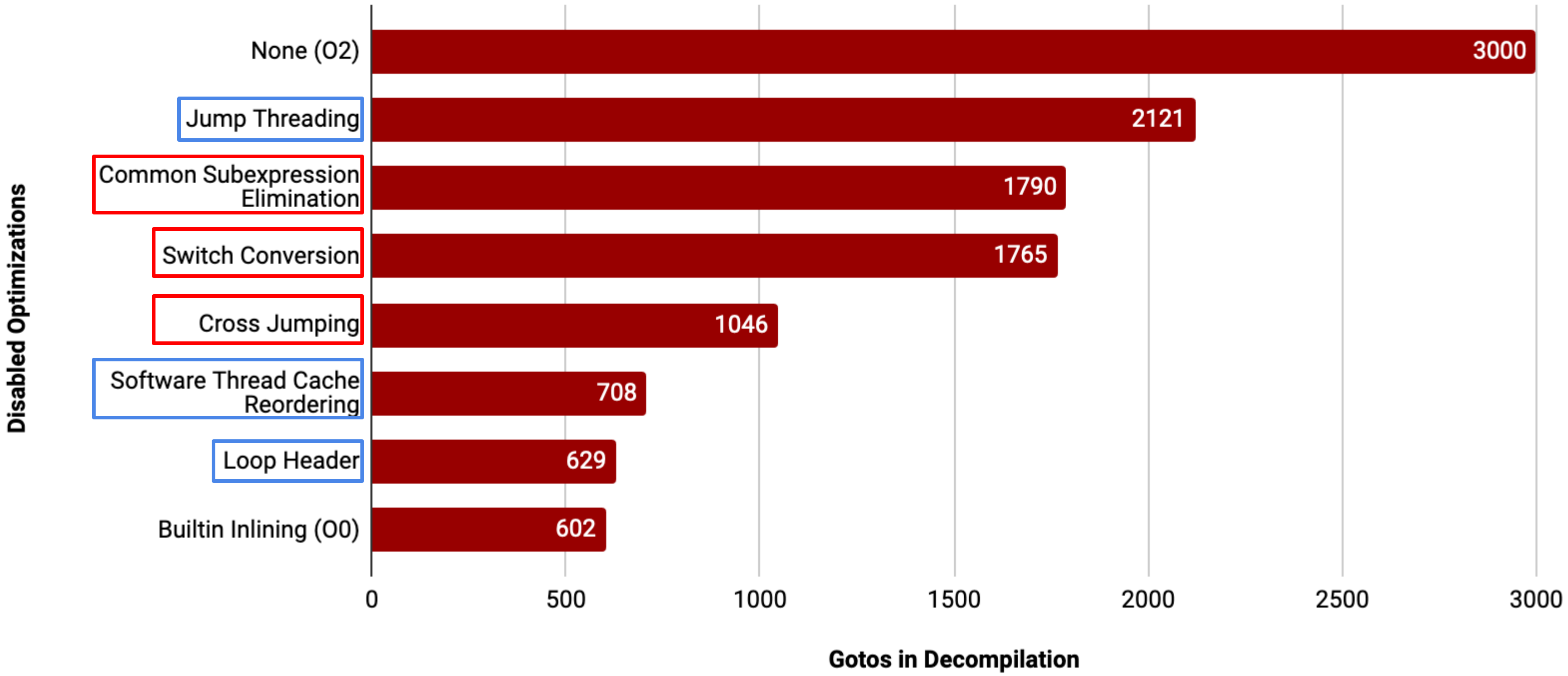
```
if (a && b)
    puts("1");
puts("2");
if (b)
    puts("3");
return 0;
```

```
if (a && b) {
    puts("1");
    puts("2");
    goto B;
}
puts("2");
if (b) {
B:
    puts("3");
}
return 0;
```

Investigating structure-breaking compiler optimizations



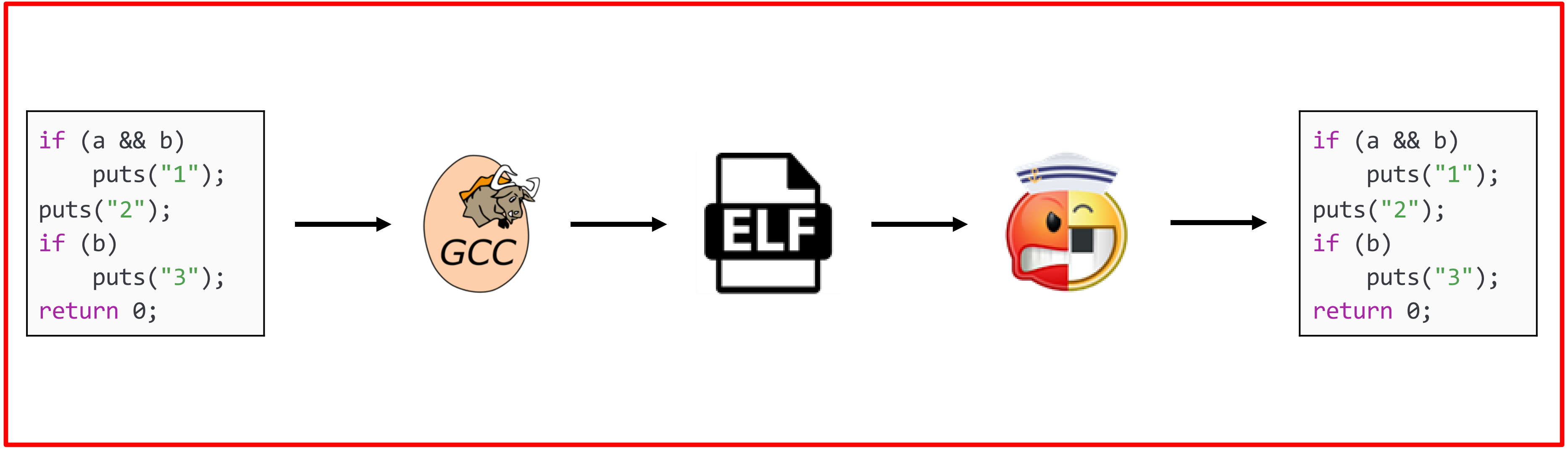
Investigating structure-breaking compiler optimizations



Investigating structure-breaking compiler optimizations

Optimization	Decompiler Effect
Jump Threading STCR Loop Header	Duplication (ISD)
CSE Switch Conversion Cross Jumping	Condensing (ISC)
Builtin Inlining Switch Lowering Non-Ret Functions	Other Transformation

Compiler-aware decompilation: SAILR

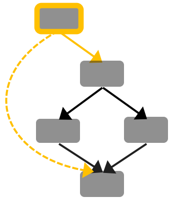


```
if (a && b)
    puts("1");
puts("2");
if (b)
    puts("3");
return 0;
```

```
if (a && b)
    puts("1");
puts("2");
if (b)
    puts("3");
return 0;
```

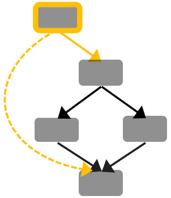
SAILR structuring algorithm

1. Define patterns to match ISD, ISC, and misc optimizations

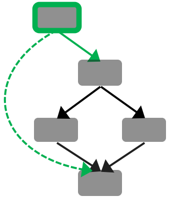


SAILR structuring algorithm

1. Define patterns to match ISD, ISC, and misc optimizations

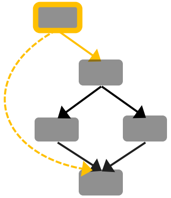


2. Identify them during structuring

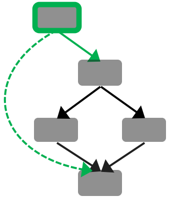


SAILR structuring algorithm

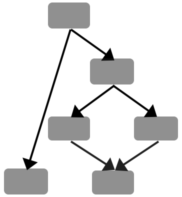
1. Define patterns to match ISD, ISC, and misc optimizations



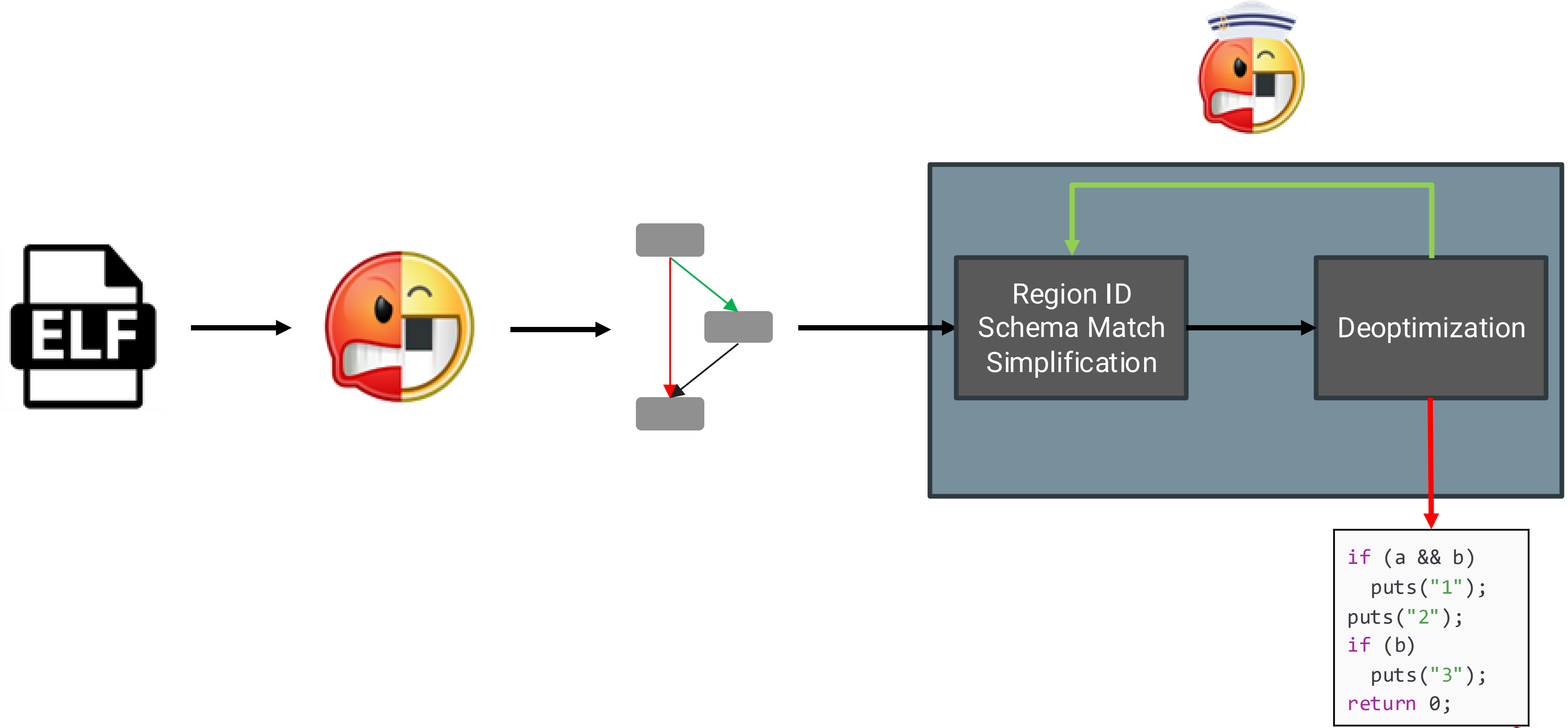
2. Identify them during structuring



3. Reverse them according to their GCC 9 implementation

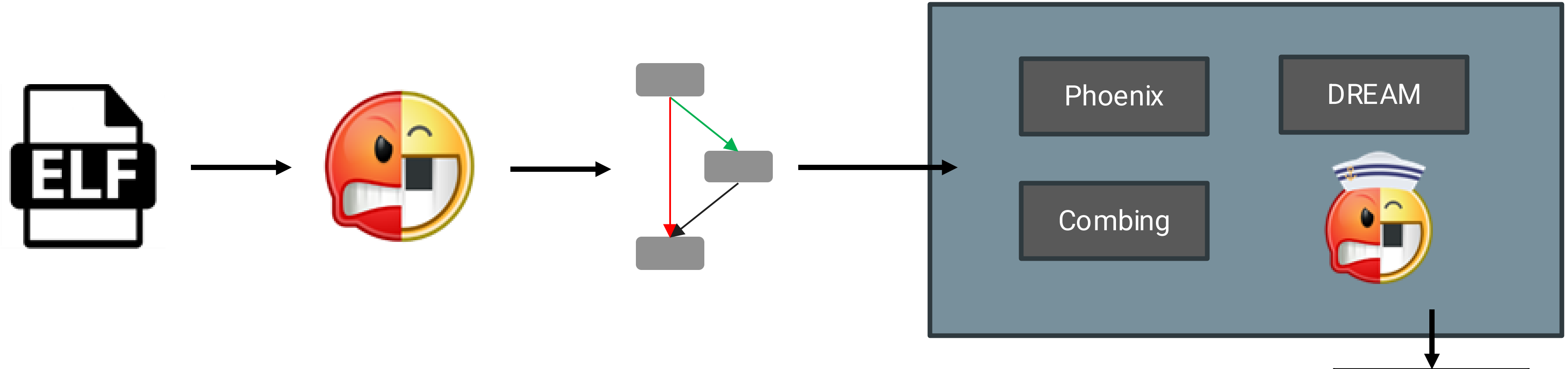


The angr decompiler & SAILR



```
if (a && b)
  puts("1");
puts("2");
if (b)
  puts("3");
return 0;
```

Reimplemented previous work



```
if (a && b)
  puts("1");
puts("2");
if (b)
  puts("3");
return 0;
```

Evaluation pipeline: source relative

```
if (a && b)
    puts("1");
puts("2");
if (b)
    puts("3");
return 0;
```

```
if (a && b) {
    puts("1");
    puts("2");
    goto B;
}
puts("2");
if (b) {
B:
    puts("3");
}
return 0;
```

Evaluation pipeline: source relative

```
if (a && b)
    puts("1");
puts("2");
if (b)
    puts("3");
return 0;
```

```
if (a && b) {
    puts("1");
    puts("2");
    goto B;
}
puts("2");
if (b) {
B:
    puts("3");
}
return 0;
```

#gotos

Evaluation pipeline: source relative

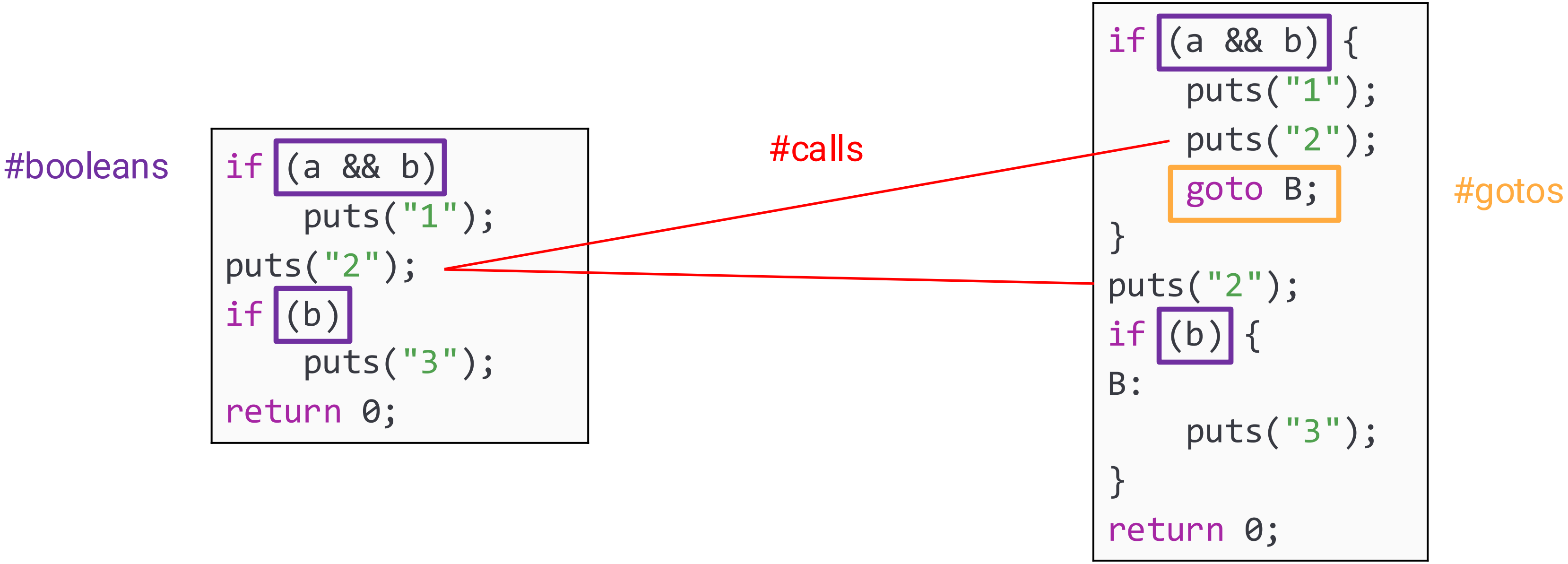
#booleans

```
if (a && b)
    puts("1");
puts("2");
if (b)
    puts("3");
return 0;
```

```
if (a && b) {
    puts("1");
    puts("2");
    goto B;
}
puts("2");
if (b) {
B:
    puts("3");
}
return 0;
```

#gotos

Evaluation pipeline: source relative



#booleans

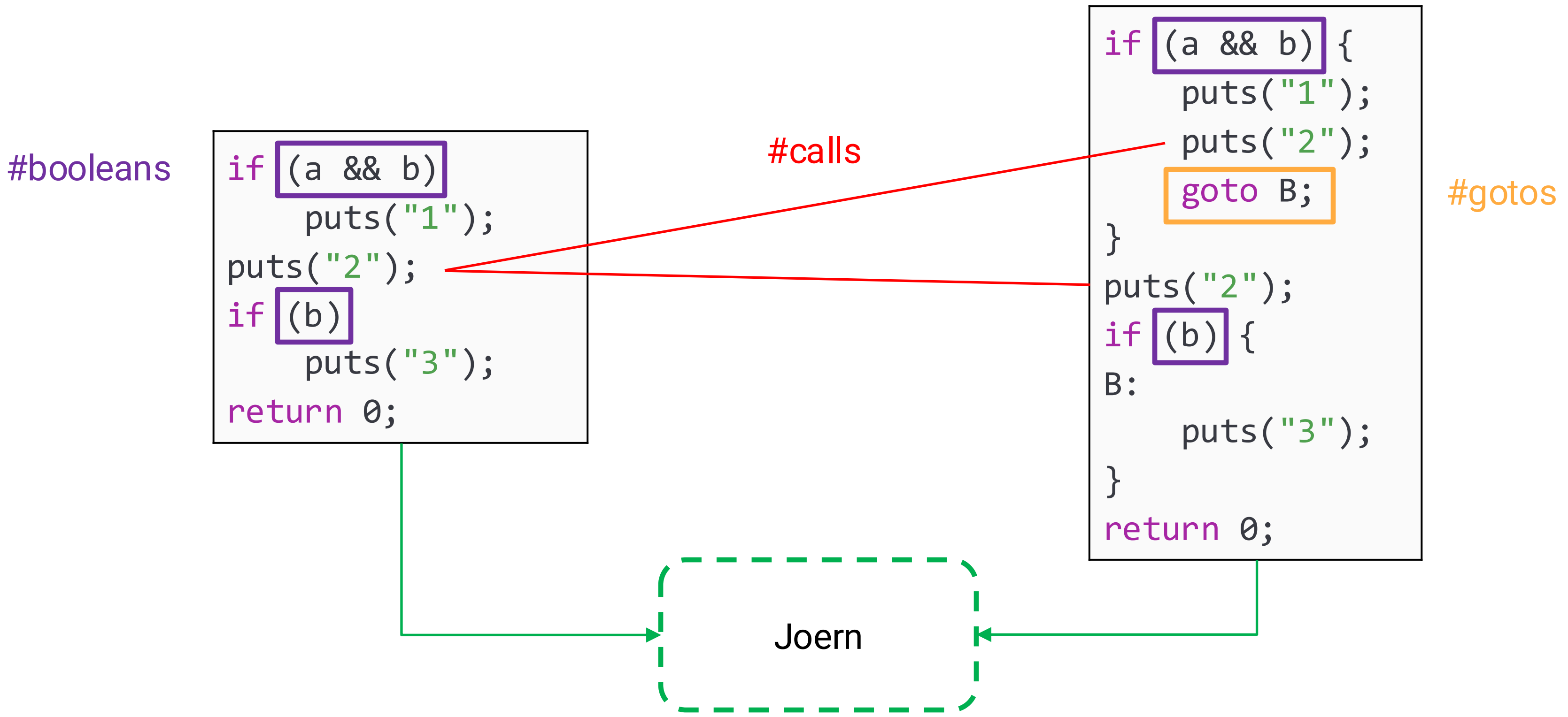
```
if (a && b)
    puts("1");
puts("2");
if (b)
    puts("3");
return 0;
```

#calls

```
if (a && b) {
    puts("1");
    puts("2");
    goto B;
}
puts("2");
if (b) {
B:
    puts("3");
}
return 0;
```

#gotos

Evaluation pipeline: source relative



#booleans

```
if (a && b)
    puts("1");
puts("2");
if (b)
    puts("3");
return 0;
```

#calls

```
if (a && b) {
    puts("1");
    puts("2");
    goto B;
}
puts("2");
if (b) {
B:
    puts("3");
}
return 0;
```

#gotos

Joern

Control Flow Graph Edit Distance (CFGED)

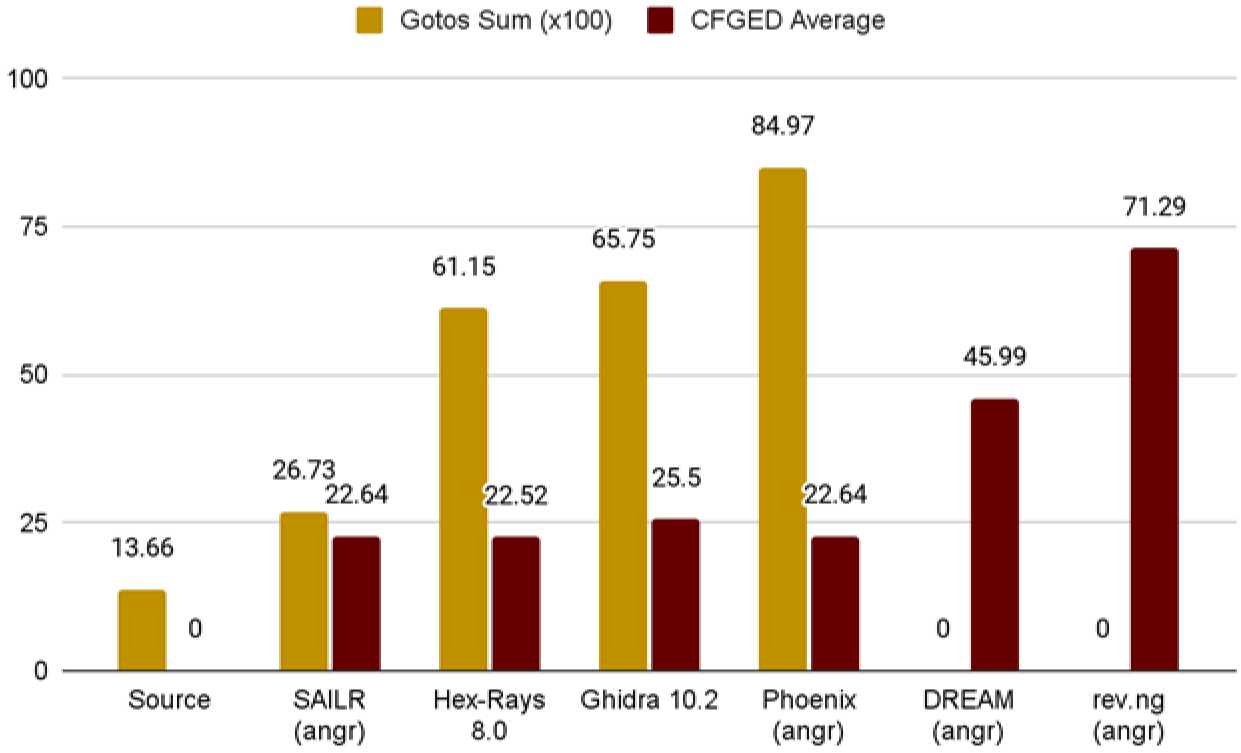
Evaluation pipeline: dataset

- **26 C Debian packages** from the top 50 more popular
  ~ 7k unique functions

- **Multiple compilers:** GCC 5,9,11; Clang 14; MSVC 14.20

- **Appendix Dataset:** malware, Linux kernel

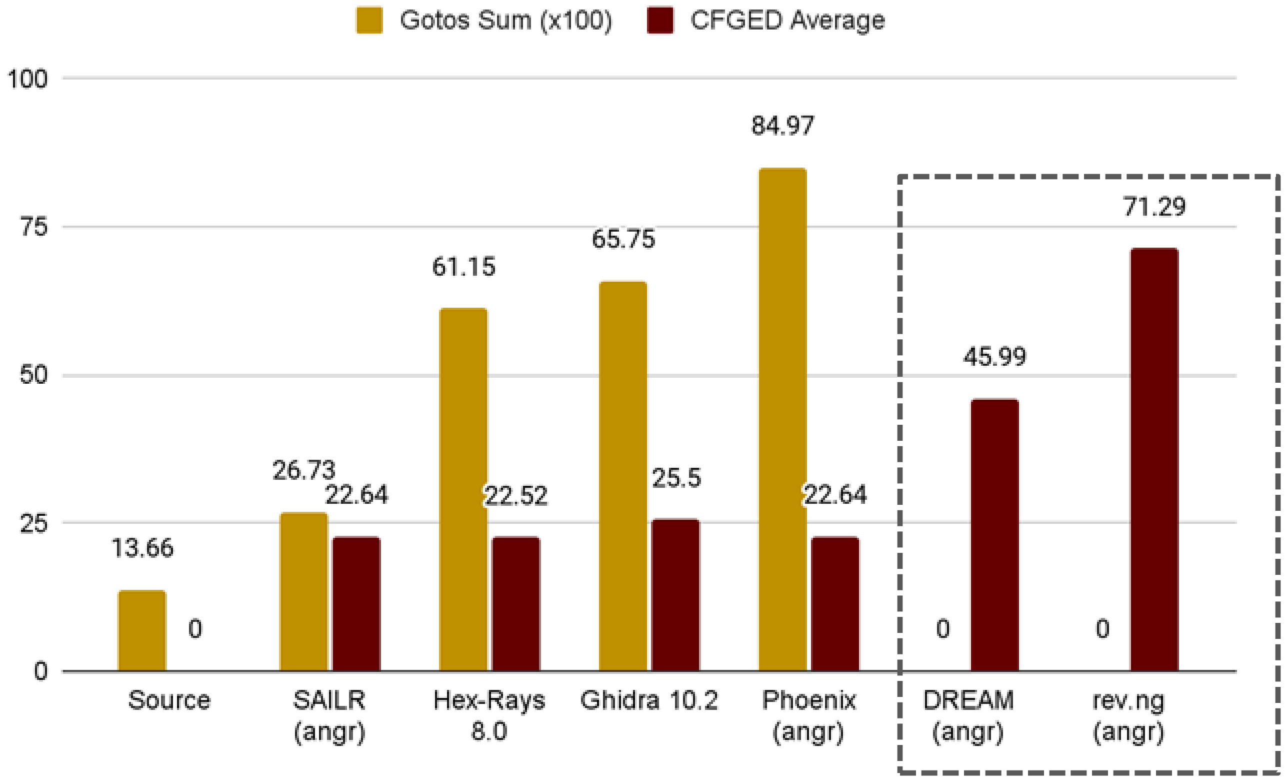
Results



Schema-based

Gotoless

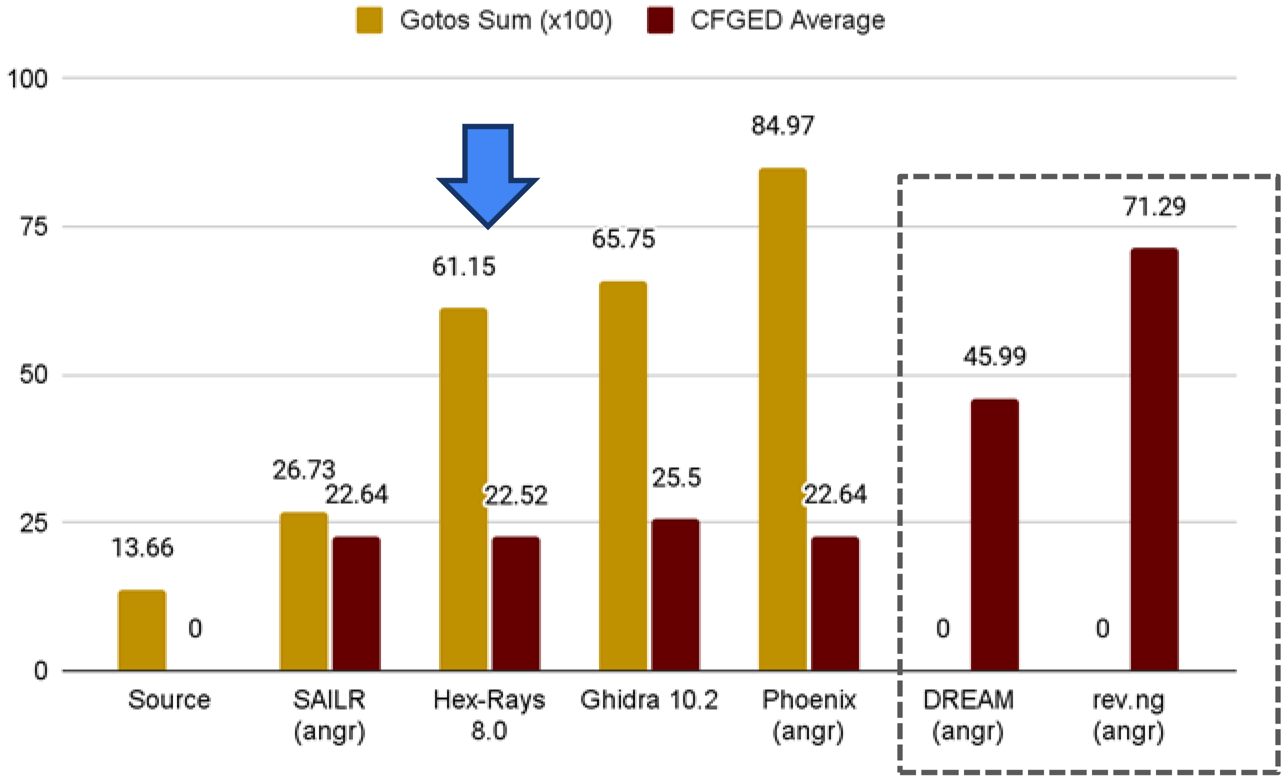
Results



Schema-based

Gotoless

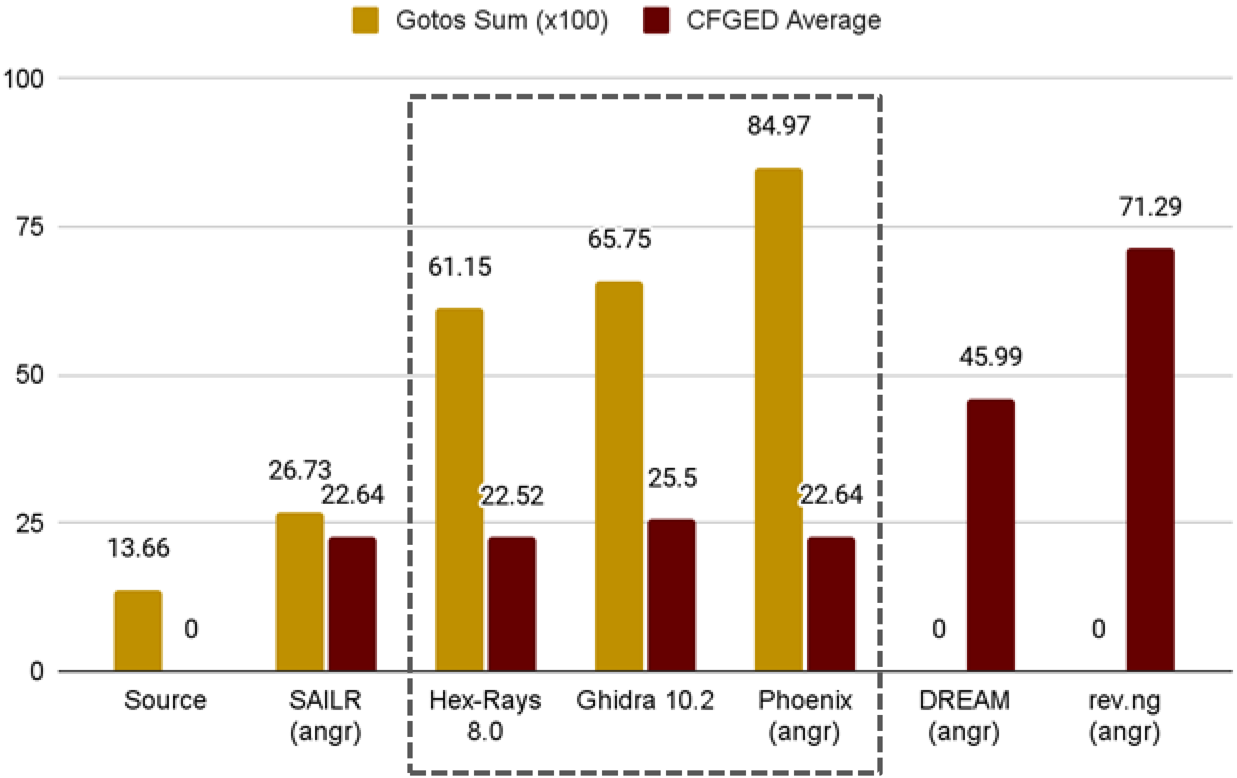
Results



Schema-based

Gotoless

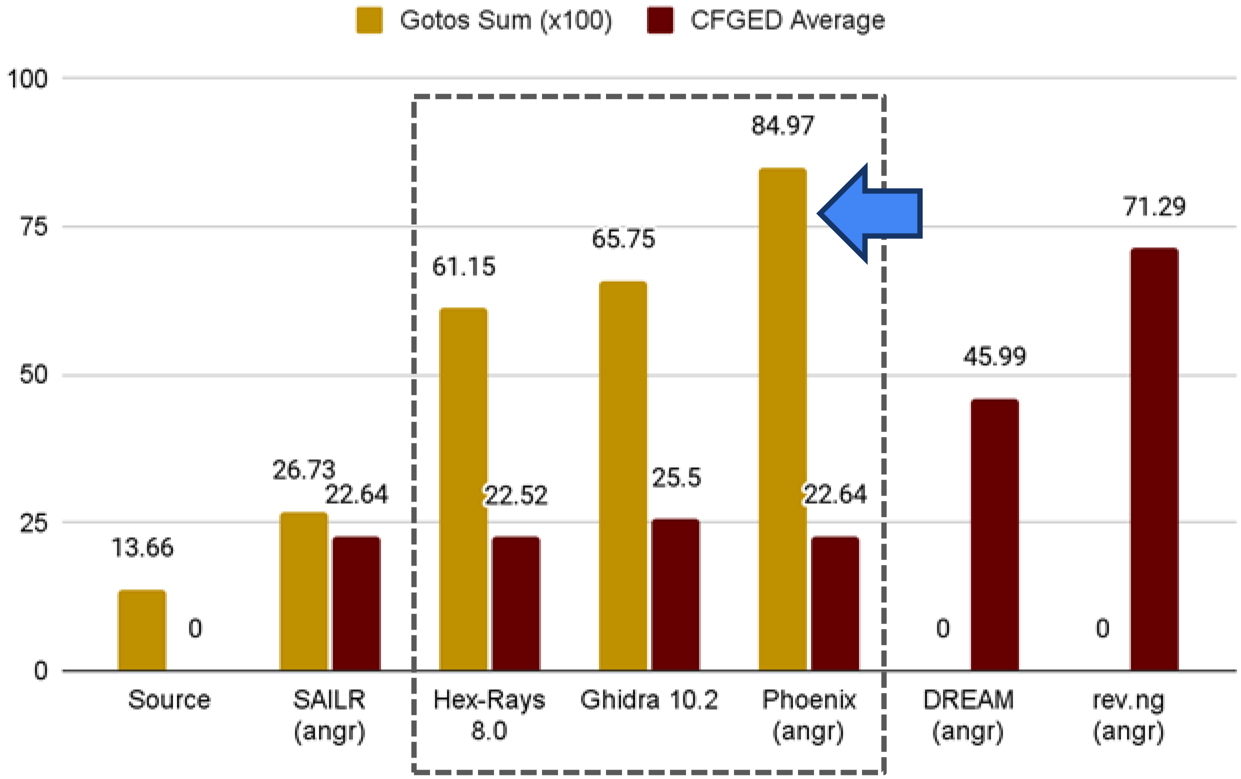
Results



Schema-based

Gotoless

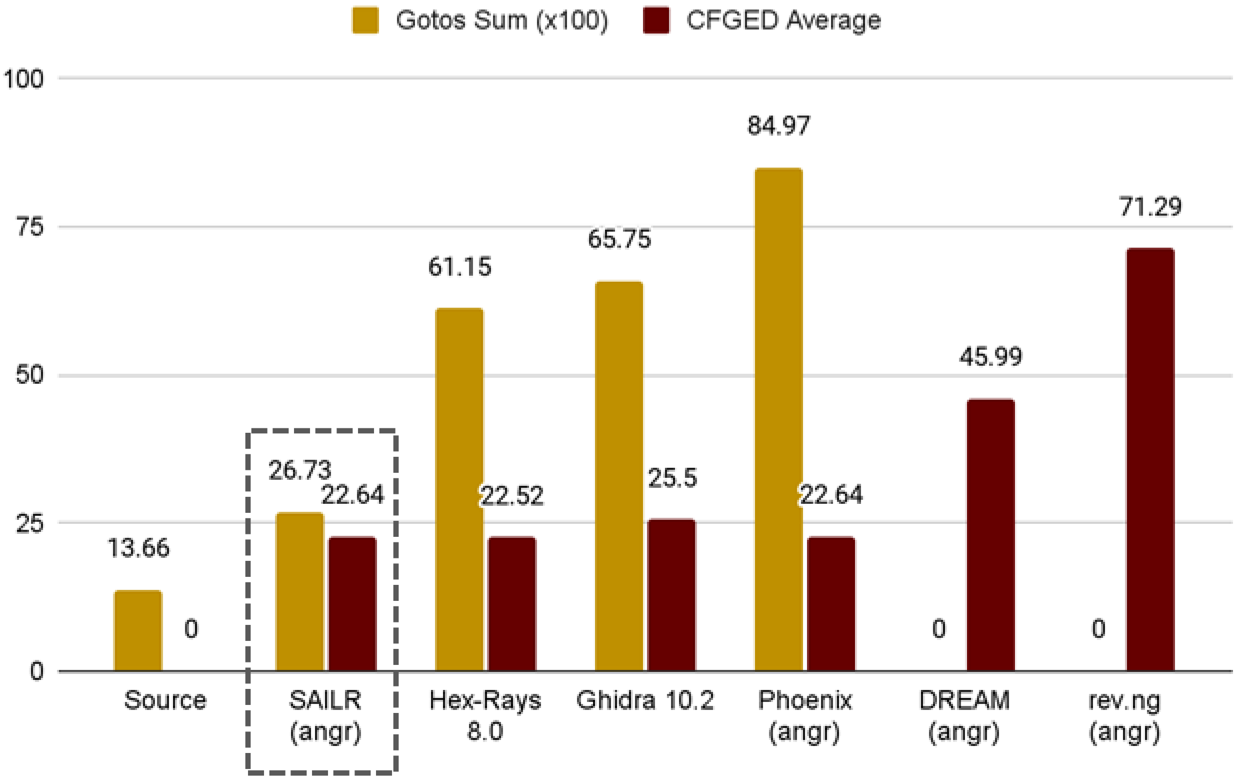
Results



Schema-based

Gotoless

Results



Schema-based

Gotoless

Results: source-like decompilation (1)

Source

```c
int schedule_job(int needs_next, int fast_job,
int mode)
{
  if (needs_next && fast_job)
  {
    complete_job();
    if (mode == EARLY_EXIT)
      goto cleanup;
    next_job();
  }
  refresh_jobs();
  if (fast_job)
    fast_unlock();
cleanup:
  complete_job();
  log_workers();
  return job_status(fast_job);
}
```

Hex-Rays (IDA Pro)

```c
int schedule_job(int a1, unsigned int a2, int
a3)
{
  if ( !a1 || !a2 )
  {
    refresh_jobs();
    if ( !a2 )
      goto LABEL_7;
    goto LABEL_5;
  }
  complete_job();
  if ( EARLY_EXIT != a3 )
  {
    next_job();
    refresh_jobs();
LABEL_5:
    fast_unlock();
  }
LABEL_7:
  complete_job();
  log_workers();
  return job_status(a2);
}
```

Results: source-like decompilation (1)

Source

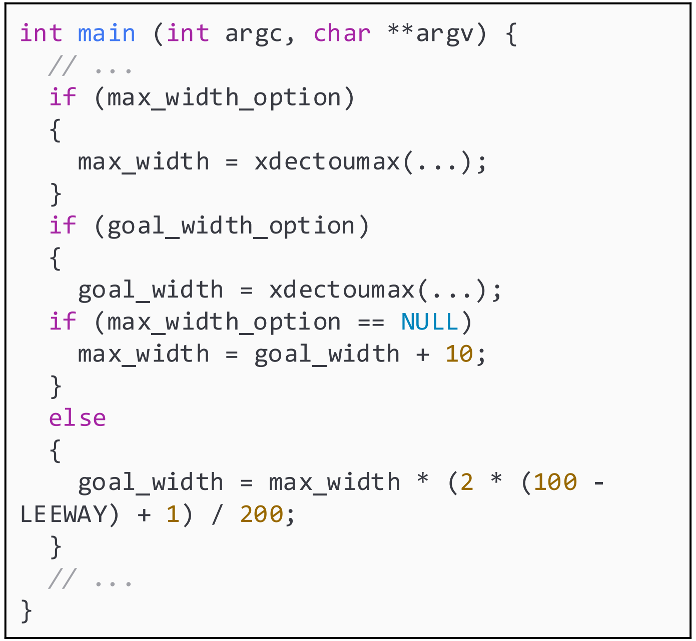
```c
int schedule_job(int needs_next, int fast_job,
int mode)
{
  if (needs_next && fast_job)
  {
    complete_job();
    if (mode == EARLY_EXIT)
      goto cleanup;
    next_job();
  }
  refresh_jobs();
  if (fast_job)
    fast_unlock();
cleanup:
  complete_job();
  log_workers();
  return job_status(fast_job);
}
```

SAILR

```c
long long schedule_job(int a0, unsigned int
a1, int a2)
{
  if (a0 && a1)
  {
    complete_job();
    if (EARLY_EXIT == a2)
      goto LABEL_4012eb;
    next_job();
  }
  refresh_jobs();
  if (a1)
    fast_unlock();
LABEL_4012eb:
  complete_job();
  log_workers();
  return job_status(a1);
}
```

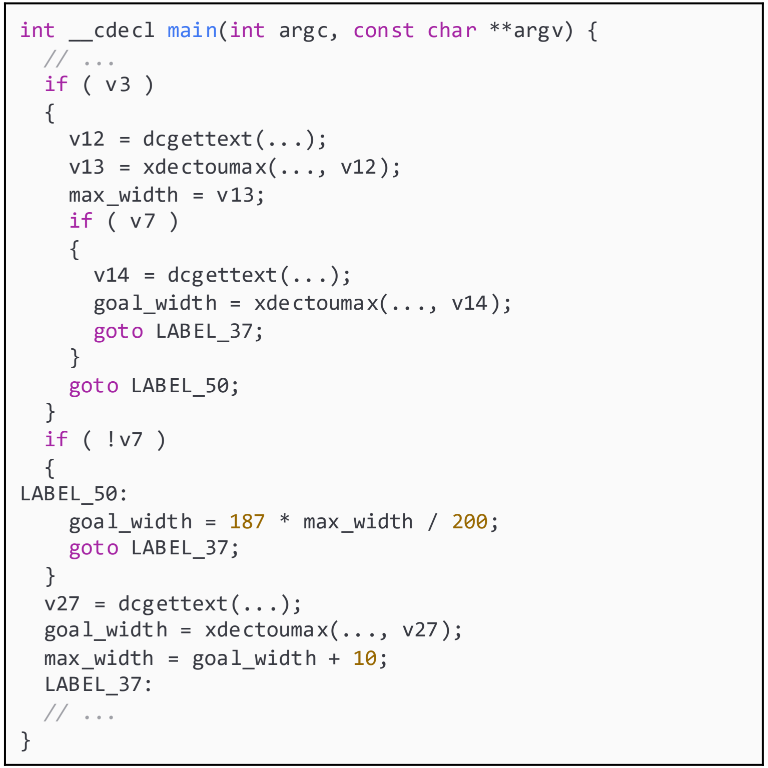
Results: source-like decompilation (2)

Source



```c
int main (int argc, char **argv) {
  // ...
  if (max_width_option)
  {
    max_width = xdectoumax(...);
  }
  if (goal_width_option)
  {
    goal_width = xdectoumax(...);
  if (max_width_option == NULL)
    max_width = goal_width + 10;
  }
  else
  {
    goal_width = max_width * (2 * (100 -
LEEWAY) + 1) / 200;
  }
  // ...
}
```

Hex-Rays (IDA Pro)



```c
int __cdecl main(int argc, const char **argv) {
  // ...
  if ( v3 )
  {
    v12 = dcgettext(...);
    v13 = xdectoumax(..., v12);
    max_width = v13;
    if ( v7 )
    {
      v14 = dcgettext(...);
      goal_width = xdectoumax(..., v14);
      goto LABEL_37;
    }
    goto LABEL_50;
  }
  if ( !v7 )
  {
LABEL_50:
    goal_width = 187 * max_width / 200;
    goto LABEL_37;
  }
  v27 = dcgettext(...);
  goal_width = xdectoumax(..., v27);
  max_width = goal_width + 10;
  LABEL_37:
  // ...
}
```

Results: source-like decompilation (2)

Source

```
int main (int argc, char **argv) {
  // ...
  if (max_width_option)
  {
    max_width = xdectoumax(...);
  }
  if (goal_width_option)
  {
    goal_width = xdectoumax(...);
  if (max_width_option == NULL)
    max_width = goal_width + 10;
  }
  else
  {
    goal_width = max_width * (2 * (100 -
LEEWAY) + 1) / 200;
  }
  // ...
}
```

SAILR

```
int main(int argc, const char **argv) {
  // ...
  if (v2)
    max_width = xdectoumax(..., dcgettext(...));
  if (!v6)
  {
    v11 = max_width * 187;
    v12 = (v11 >> 31 CONCAT v11) /m 200;
    v13 = v12 / 0x100000000;
    goal_width = v12;
  }
  else
  {
    goal_width = xdectoumax(..., dcgettext(...));
    if (!v2)
      max_width = goal_width + 10;
  }
  // ...
}
```

Results: source-like decompilation (3)

Source

```c
int main (int argc, char **argv) {
  // ...
  switch (optchar) {
  default:
    // ...
  case 'c':
    // ...
  case 's':
    // ...
  case 't':
    // ...
  case 'u':
    // ...
  case 'w':
    // ...
  case 'g':
    // ...
  case 'p':
    // ...
  case_GETOPT_HELP_CHAR;
    // ...
  case_GETOPT_VERSION_CHAR(...);
    // ...
  }
  // ...
}
```

Hex-Rays (IDA Pro)

```c
int __cdecl main(int argc, const char **argv, const char **envp) {
  // ...
  if ( v8 == 112 )
  {
    // ...
  }
  else if ( v8 <= 112 )
  {
    if ( v8 == -130 )
      // ...
    if ( v8 <= -130 )
    {
      if ( v8 == -131 )
      {
        // ...
      }
LABEL_53:
      // ...
    }
    if ( v8 == 99 )
    {
      // ...
    }
    else
    {
      if ( v8 != 103 )
        goto LABEL_53;
      // ...
    }
  }
  else if ( v8 == 116 )
  {
    // ...
  }
  else if ( v8 <= 116 )
  {
    if ( v8 != 115 )
      goto LABEL_53;
    // ...
  }
  else if ( v8 == 117 )
  {
    // ...
  }
  else
  {
    if ( v8 != 119 )
      goto LABEL_53;
    // ...
  }
  // ...
}
```

Results: source-like decompilation (3)

Source

```
int main (int argc, char **argv) {
  // ...
  switch (optchar) {
  default:
    // ...
  case 'c':
    // ...
  case 's':
    // ...
  case 't':
    // ...
  case 'u':
    // ...
  case 'w':
    // ...
  case 'g':
    // ...
  case 'p':
    // ...
  case_GETOPT_HELP_CHAR;
    // ...
  case_GETOPT_VERSION_CHAR(...);
    // ...
  }
  // ...
}
```

SAILR

```
int main(int argc, const char **argv) {
  // ...
  switch (v5) {
  case 112:
    // ...
  case 116:
    // ...
  case 4294967166:
    // ...
  case 117:
    // ...
  case 115:
    // ...
  case 99:
    // ...
  case 4294967165:
    // ...
  case 119:
    // ...
  case 103:
    // ...
  default:
    // ...
  }
  // ...
}
```

Takeaways

1. Compilers play a central role in decompilation and should be understood to improve decompilation

2. Decompilers should reduce structure failures (spurious gotos) without significantly affect graph edit distance

3. One way to achieve source-exact decompilation is to revert optimizations

4. Compilers implement similar optimizations, making deoptimizations generic

Thank you

Zion Leonahenahe Basque
✉: zbasque@asu.edu
𝕏: @mahal0z
: https://github.com/angr/angr
🔬: https://github.com/mahaloz/sailr-eval

```
pip install angr
angr decompile /bin/true
```