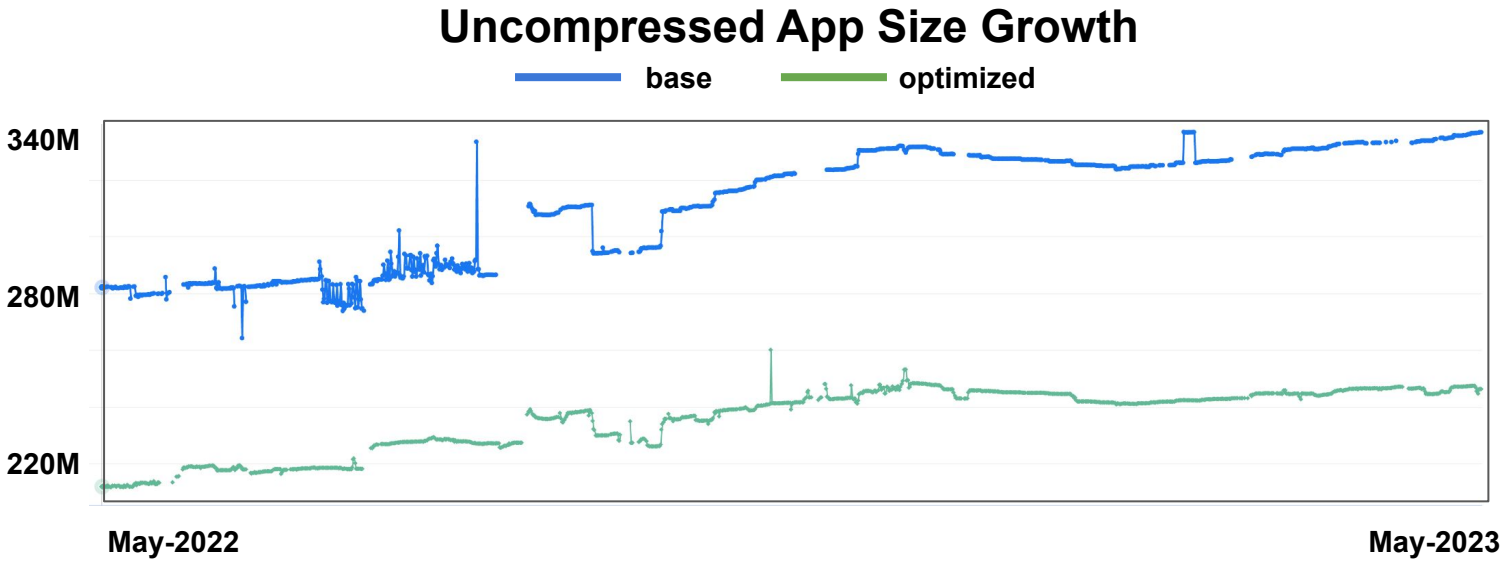
Advances in Function Merging and Symbolication

Alex Borcan and Kyungwoo Lee

Meta

App Size Continues to Grow

- Large and slow apps impact user experience and user retention
- Code size optimizations (e.g, Function Merging) are critical!

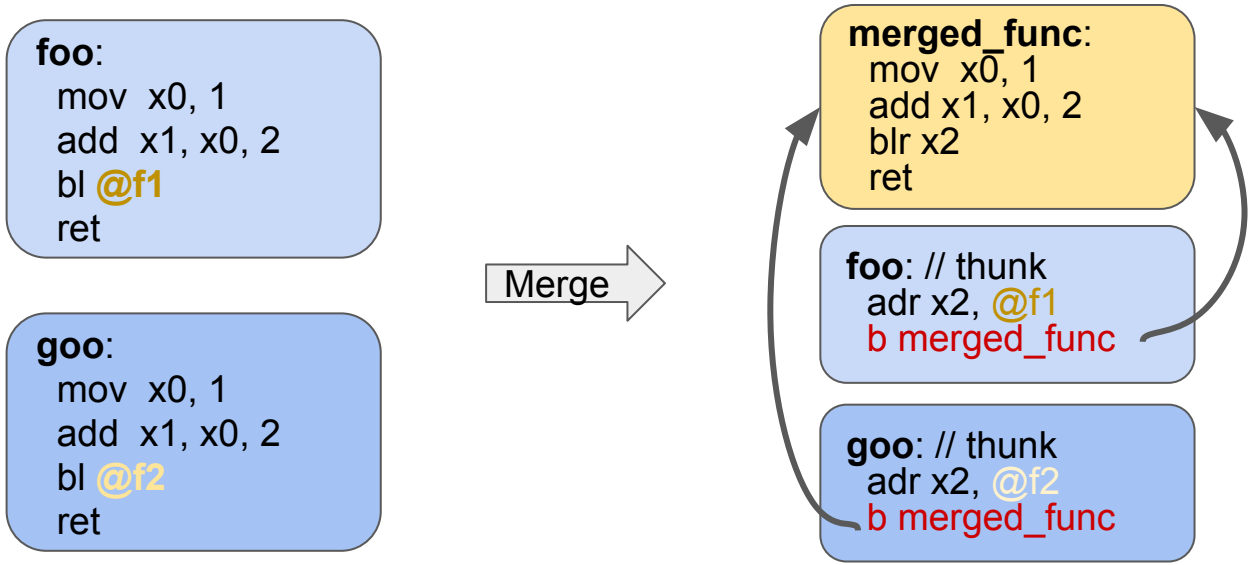


Agenda

- Global Function Merger
  - Leverages the linker’s Identical Code Folding (ICF)
- Improving Safe ICF with Thunks
- Symbolication of Merged Functions

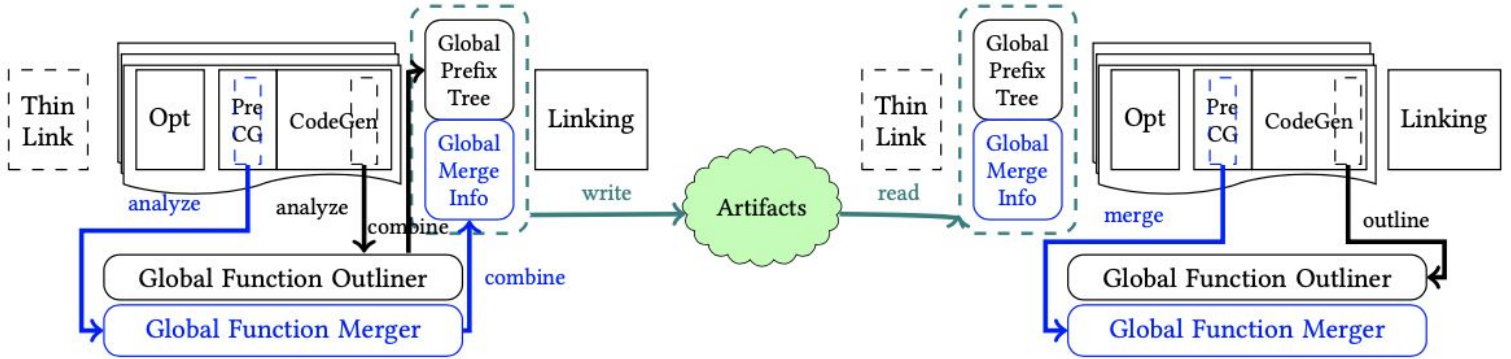
Traditional Function Merger

- Merges identical functions, similar to the linker's ICF.
- Swift's merger extends this by merging similar functions through parameterization.



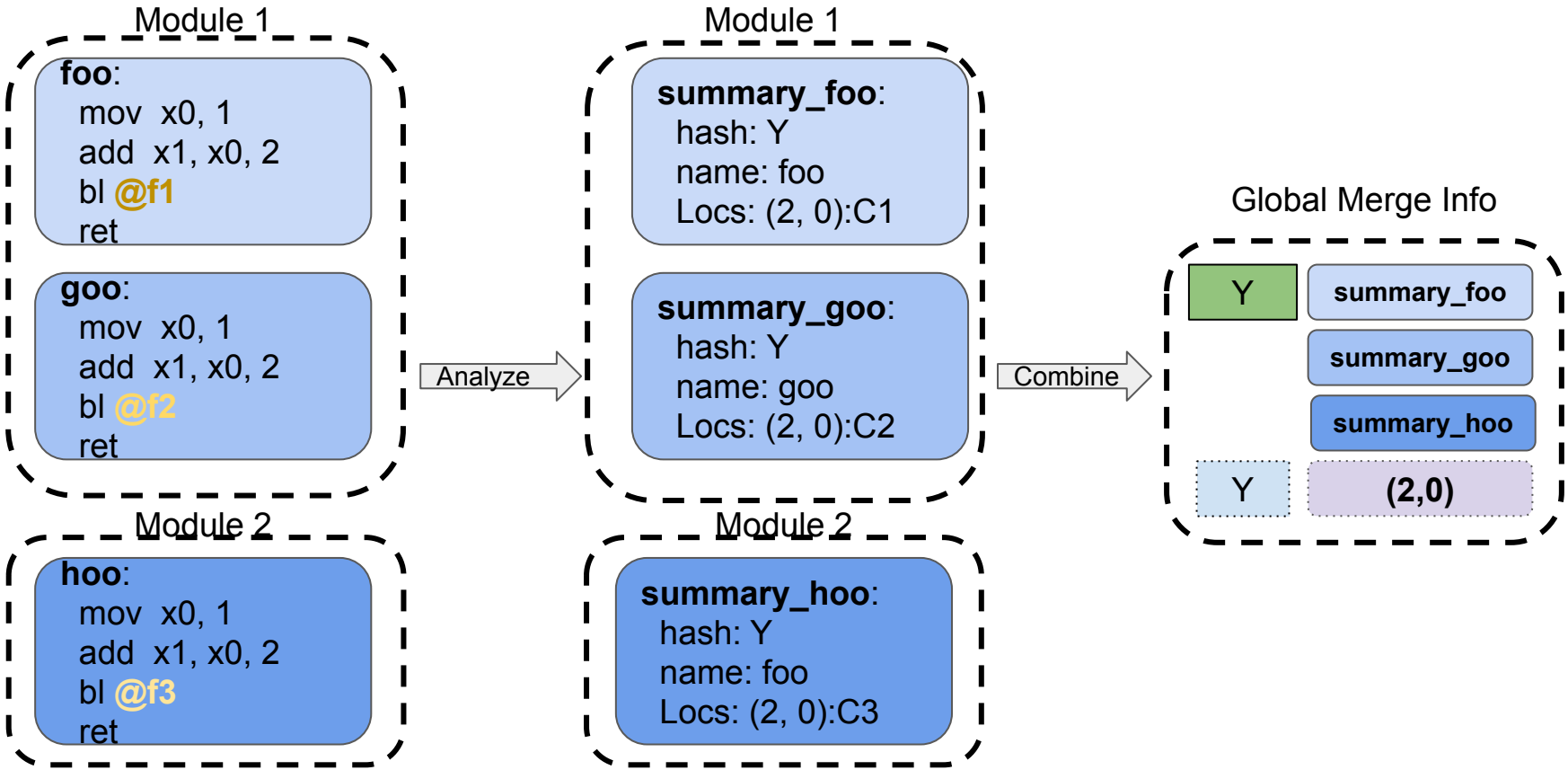
Global Function Merger

- Use the CodenGen data framework
  - Initially, implemented for the Global Function Outliner
  - Writer/Gen: Generates a codegen data summary for each module.
  - Reader/Use: Read the combined codegen data to optimistically create merging instance
- The actual merging is performed via ICF during link time.

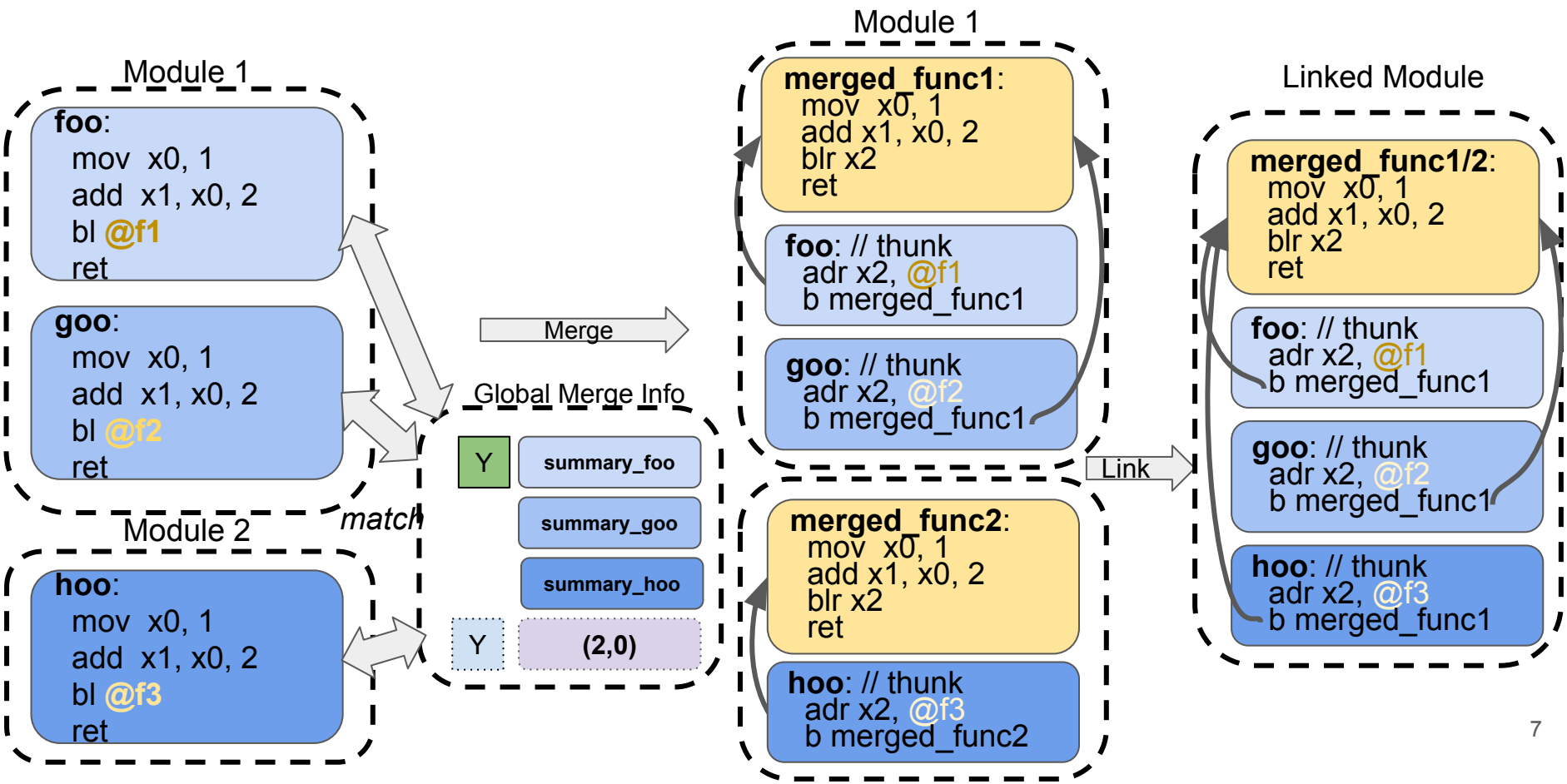


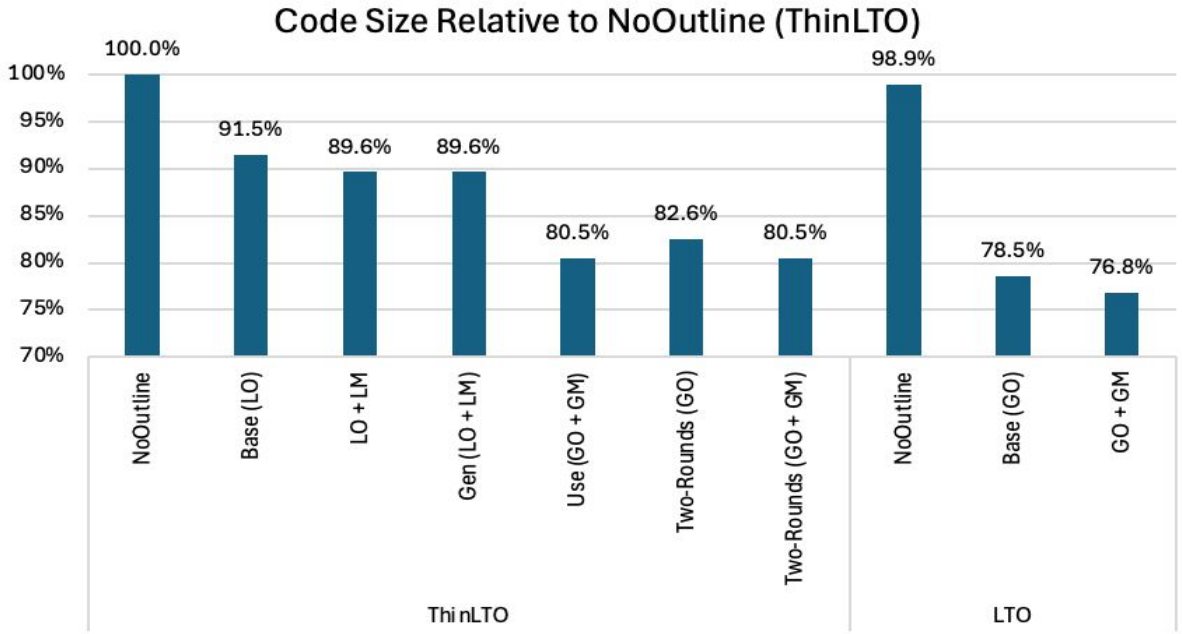
(a)

ThinLTO + Global Function Merger (1st CG)



ThinLTO + Global Func Merger (2nd CG)





LO/GO: Local/Global Outlining, ***-mllvm -enable-machine-outliner=always***

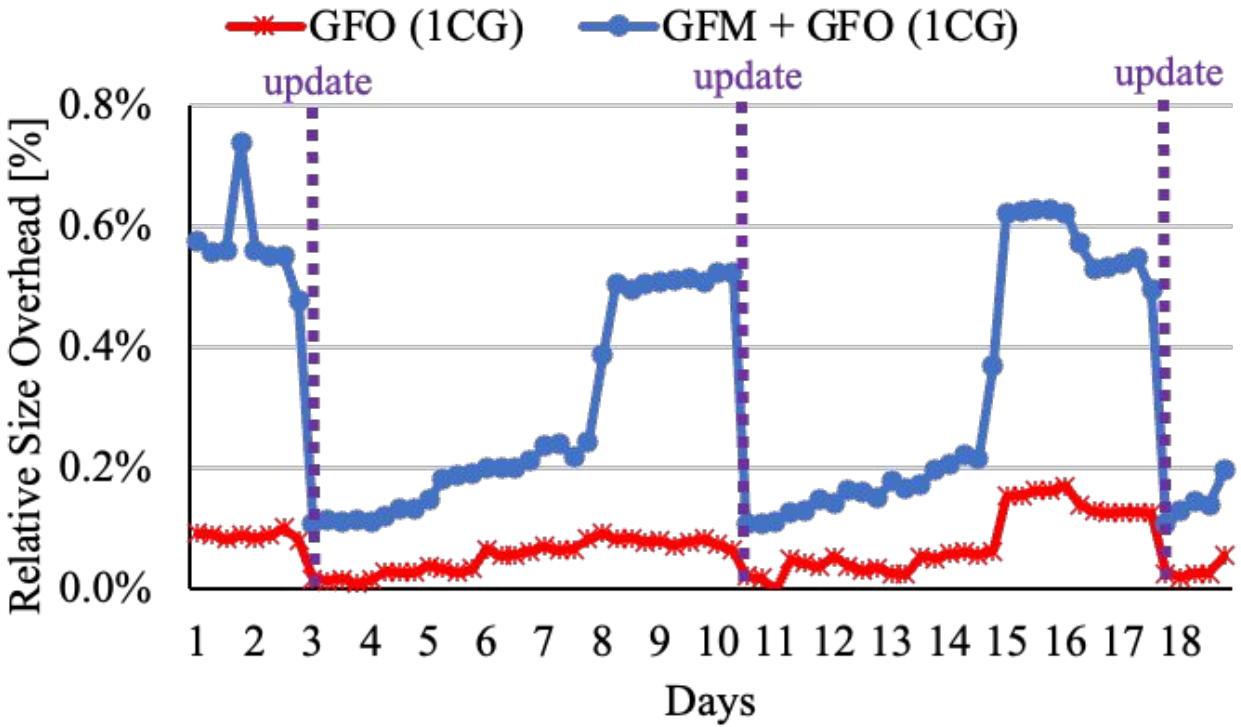
LM/GM: Local/Global Merging, ***-mllvm -enable-global-merge-func***

Gen: Generate codegen data, ***-fcodegen-data-generate={path}***

Use: Use codegen data, ***-fcodegen-data-use={path}***

Two-Rounds: Repeat CG for Gen and Use in place, ***-mllvm -codegen-data-thinlto-two-rounds***

Weekly Codegen Data: Size Overhead Impact



Improving Linker ICF with thunks

Meta

main.cpp

```
NO_INLINE
int a(int aa){
  return ++aa;
}

NO_INLINE
int b(int bb){
  return ++bb;
}

int main(){
  return
      &a==&b;
}
```

main.o

```
a(int):
add   w0, #1
ret

b(int):
add   w0, #1
ret

main:
mov   w0, #1
addr  x1, a
addr  x2, b
cmp   x1, x2
set   x0, eq
ret
```

--icf=all

```
a(int):
add   w0, #1
ret

main:
mov   w0, #1
addr  x1, a
addr  x2, a
cmp   x1, x2
set   x0, eq
ret
```

--icf=safe_thunks

```
a(int):
add   w0, #1
ret

b_thunk:
b     a(int)

main:
mov   w0, #1
addr  x1, a
addr  x2, b_t
cmp   x1, x2
set   x0, eq
ret
```

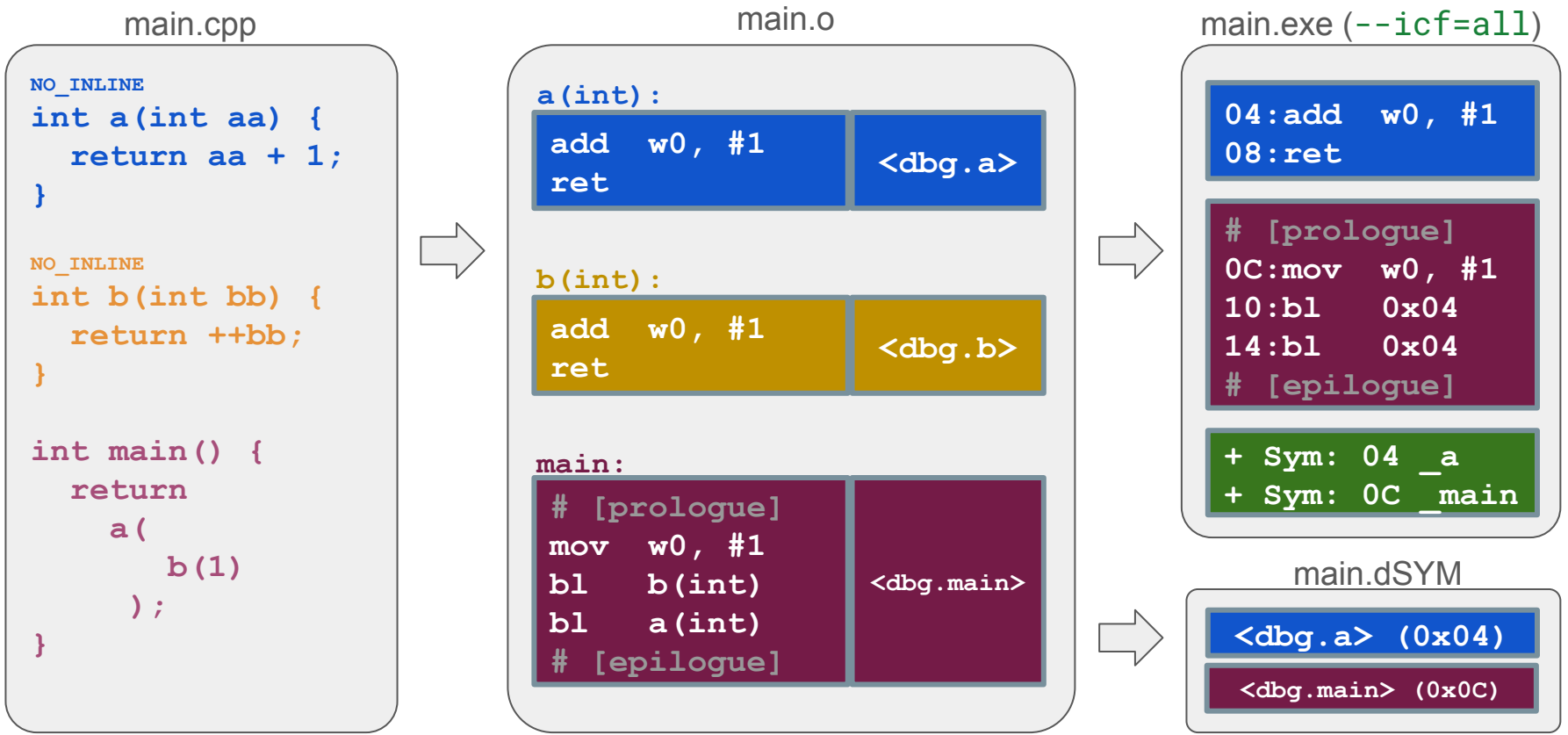
Thunks ICF: Details & Enabling

Meta

- Shrinks binary size by ~0.45% compared to regular “safe” ICF (`--icf=safe`)
- Upstreamed for LLD MachO ARM64 linker
- Enabled by using `--icf=safe_thunks` when linking

  *Note: Other ICF modes are:
  - `--icf=none` : Disables identical function merging
  - `--icf=all` : Enables merging for all identical functions
  - `--icf=safe` : Enables merging of non-address-significant functions (identified via the `__llvm_addrsig` section).

Function Merging And Debug Information



Function Merging: Symbolication Issues

Meta

main.cpp

```
NO_INLINE
int a(int aa) {
  return aa + 1;
}

NO_INLINE
int b(int bb) {
  return ++bb;
}

int main() {
  return
    a(
      b(1)
    );
}
```

main.exe

```
04:add  w0, #1
08:ret
```

```
# [prologue]
0C:mov  w0, #1
10:bl   0x04
14:bl   0x04
# [epilogue]
```

```
+ Sym: 04 _a
+ Sym: 0C _main
```

main.dSYM

```
<dbg.a> (0x04)
```

```
<dbg.main> (0x0C)
```

Confusing call stacks

crash in a():

```
0x04 a()    :main.cpp:2
0x10 main() :main.cpp:11
```

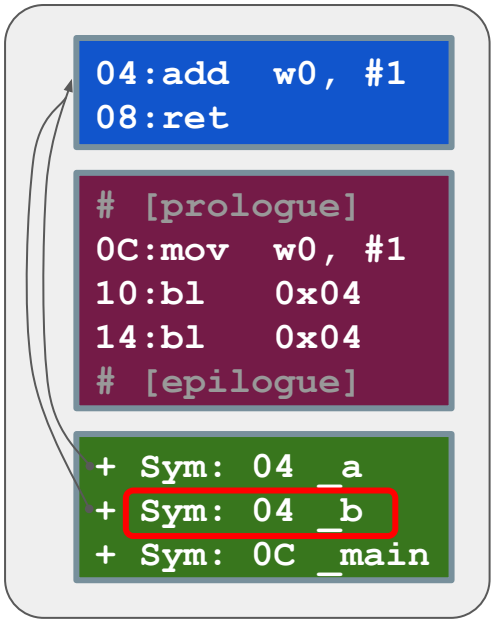
crash in b():

```
0x04 a()    :main.cpp:2
0x14 main() :main.cpp:12
```

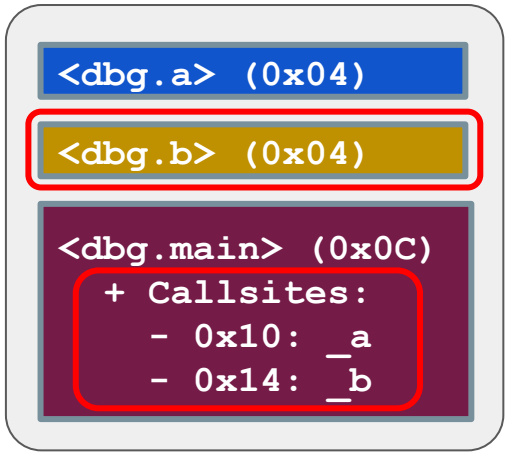
Merged Function Symbolication: More Data Is Needed

Meta

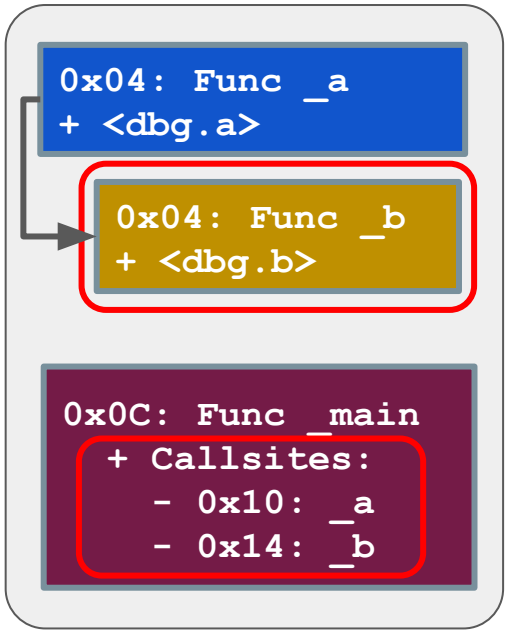
main.exe
(+ keep merged symbols)



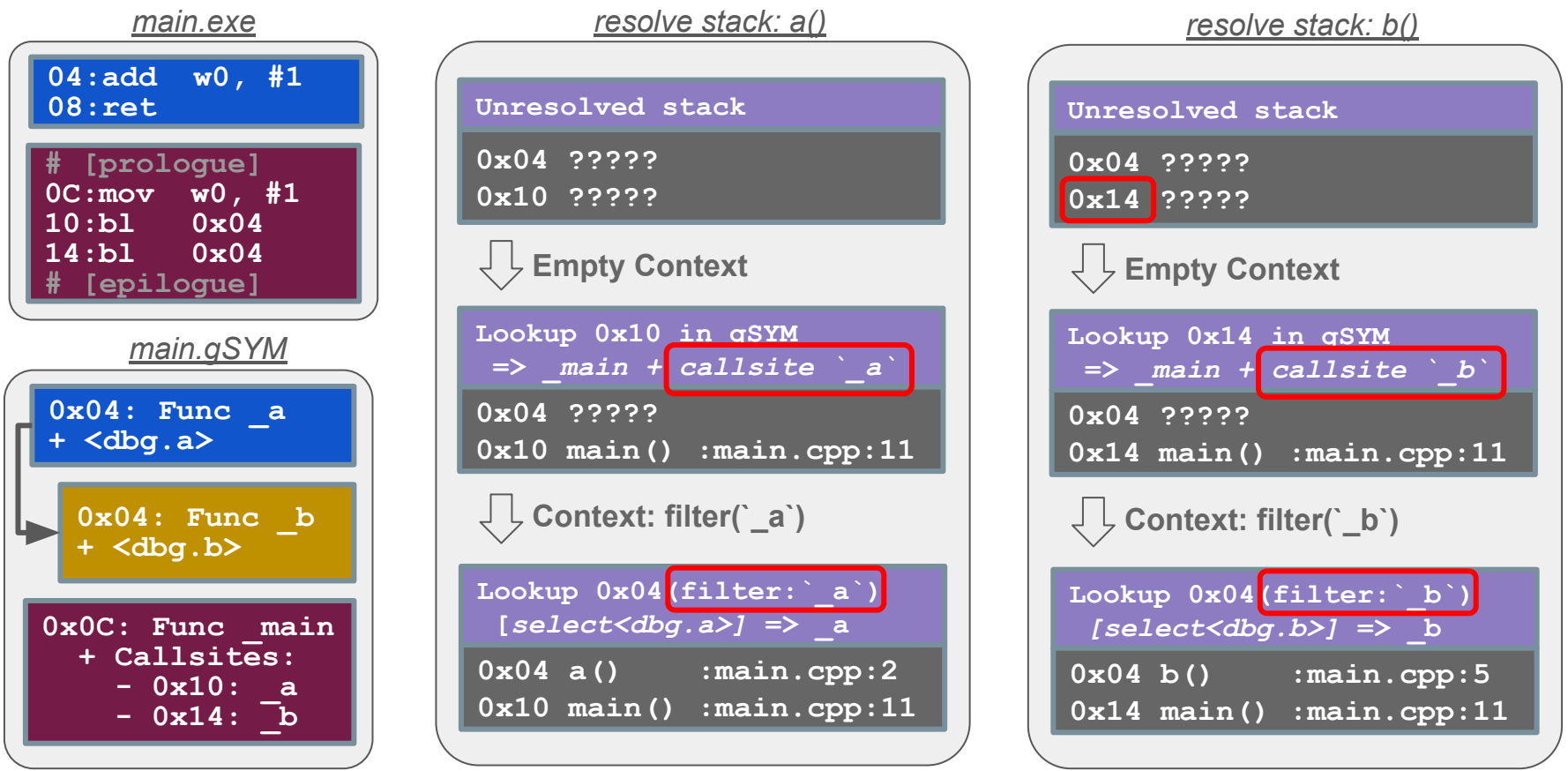
main.dSYM
(+keep **merged debug info** and **callsites**)



main.gSYM
(+keep **merged debug info** and **callsites**)



Merged Function: Accurate Symbolication with gSYM



Merged Function Symbolication: Details & Enabling

Meta

- Currently upstream supports merged function symbolication via gSYM for MachO
- Possible to extend debuggers and other symbolication pipelines to take advantage of the additional debug data
- Enable data generation via:
  - clang: `-mllvm -emit-func-debug-line-table-offsets`
    `* Ensure debug info contains callsite info`
  - lld: `--keep-icf-stabs`
  - dsymutil: `* Ensure using Mar_25_2025+ version`
  - llvm-gsymutil: `--merged-functions --dwarf-callsites`

- Symbolicate traces via:
  - llvm-gsymutil: `--merged-functions`
    `--merged-functions-filter="<FILTER>"`

Questions ?

[1] Enhanced Machine Outliner – Part 2: ThinLTO/NoLTO,
https://discourse.llvm.org/t/rfc-enhanced-machine-outliner-part-2-thinlto-nolto/78753
[2] Global Function Merging, https://discourse.llvm.org/t/rfc-global-function-merging/82608
[3] Enhance safe ICF with thunk-based deduplication, https://github.com/llvm/llvm-project/pull/106573
[4] New DWARF attribute for symbolication of merged functions,
https://discourse.llvm.org/t/rfc-new-dwarf-attribute-for-merged-functions/79434
[5] Extending gSYM Format with Call Site Information for Merged Function Disambiguation,
https://discourse.llvm.org/t/rfc-extending-gsym-format-with-call-site-information/80682
[6] Supporting ICF-Merged Functions in GSYM Debug Format,
https://discourse.llvm.org/t/rfc-supporting-icf-merged-functions-in-gsym/80292

Meta