Selfie: A Sandbox for Principled Systems Engineering

Christoph Kirsch, University of Salzburg, Austria

ARM Summit 2017, Cambridge, UK

Joint Work

- Alireza Abyaneh
- Martin Aigner
- Sebastian Arming
- Christian Barthel
- Thomas Hütter
- Michael Lippautz
- Cornelia Mayer
- Simone Oblasser

Inspiration

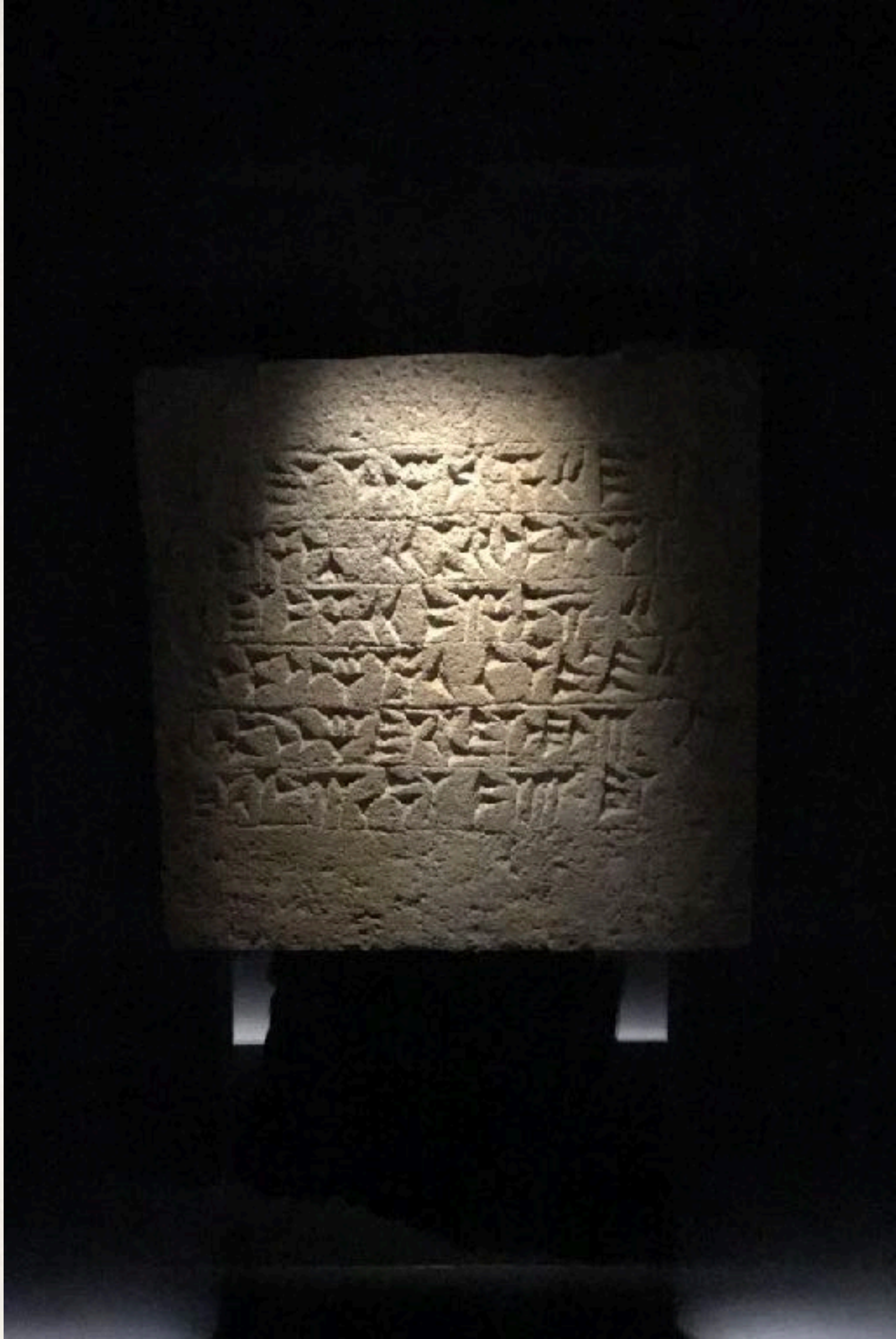
- Armin Biere: SAT/SMT Solvers
- Donald Knuth: Art
- Jochen Liedtke: Microkernels
- David Patterson: RISC
- Niklaus Wirth: Compilers

Teaching Computer Science from First Principles!

…with research
as side effect!

What is the meaning of this sentence?

Selfie as in self-referentiality



Interpretation

Compilation

Teaching the Construction of Semantics of Formalisms

Virtualization

Verification

Selfie: Teaching Computer Science [selfie.cs.uni-salzburg.at]

- *Selfie* is a self-referential 7k-line C implementation (in a single file) of:

1. a self-compiling compiler called ***starc*** that compiles a tiny subset of C called C Star (C*) to a tiny subset of MIPS32 called MIPSter,

2. a self-executing emulator called ***mipster*** that executes MIPSter code including itself when compiled with starc,

3. a self-hosting hypervisor called ***hypster*** that virtualizes mipster and can host all of selfie including itself,

4. a tiny C* library called ***libcstar*** utilized by all of selfie, and

5. a tiny, experimental SAT solver called ***babysat***.

Also, there is a…

- linker (in-memory only)
- disassembler (w/ source code line numbers)
- debugger (tracks full machine state)
- profiler (#proc-calls, #loop-iterations, #loads, #stores)
- RISC-V support (separate branch on github)

Discussion of Selfie recently reached 3rd place on Hacker News

news.ycombinator.com

Website

selfie.cs.uni-salzburg.at

Book (Draft)

leanpub.com/selfie

Code

github.com/cksystemsteaching/selfie

“Selfie and the Basics”

Onward! 2017 Paper @ SPLASH in Vancouver nsf.gov/csforall

code.org

computingatschool.org.uk

programbydesign.org

bootstrapworld.org

k12cs.org

csfieldguide.org.nz

5 statements:
assignment
while
if
return
procedure()

no data types other than int and int* and dereferencing: the * operator

```
int atoi(int *s) {
    int i;
    int n;
    int c;

    i = 0;
    n = 0;
    c = *(s+i);

    while (c != 0) {
        n = n * 10 + c - '0';
        if (n < 0)
            return -1;

        i = i + 1;
        c = *(s+i);
    }

    return n;
}
```

character literals
string literals

integer arithmetics
pointer arithmetics

no bitwise operators
no Boolean operators

library: exit, malloc, open, read, write

Scarcity versus Abundance

If you want structs implement them!

```
> make
cc -w -m32 -D'main(a,b)=main(a,char**argv)' selfie.c -o selfie
```

bootstrapping `selfie.c` *into x86* `selfie` *executable*
using standard C compiler

(also available for RISC-V machines)

```
> ./selfie
./selfie: usage: selfie { -c { source } | -o binary | -s assembly
| -l binary } [ ( -m | -d | -y | -min | -mob ) size ... ]
```

selfie usage

```
> ./selfie -c selfie.c

./selfie: this is selfie's starc compiling selfie.c

./selfie: 176408 characters read in 7083 lines and 969 comments
./selfie: with 97779(55.55%) characters in 28914 actual symbols
./selfie: 261 global variables, 289 procedures, 450 string literals
./selfie: 1958 calls, 723 assignments, 57 while, 572 if, 243 return
./selfie: 121660 bytes generated with 28779 instructions and 6544
bytes of data
```

compiling `selfie.c` *with x86* `selfie` *executable*

(takes seconds)

```
> ./selfie -c selfie.c -m 2 -c selfie.c

./selfie: this is selfie's starc compiling selfie.c

./selfie: this is selfie's mipster executing selfie.c with 2MB of
physical memory

selfie.c: this is selfie's starc compiling selfie.c

selfie.c: exiting with exit code 0 and 1.05MB of mallocated memory

./selfie: this is selfie's mipster terminating selfie.c with exit code
0 and 1.16MB of mapped memory
```

compiling `selfie.c` *with x86* `selfie` *executable into a MIPSter executable*

and

then running that MIPSter executable to compile `selfie.c` *again*

(takes ~6 minutes)

```
> ./selfie -c selfie.c -o selfie1.m -m 2 -c selfie.c -o selfie2.m

./selfie: this is selfie's starc compiling selfie.c
./selfie: 121660 bytes with 28779 instructions and 6544 bytes of data
written into selfie1.m

./selfie: this is selfie's mipster executing selfie1.m with 2MB of
physical memory

selfie1.m: this is selfie's starc compiling selfie.c
selfie1.m: 121660 bytes with 28779 instructions and 6544 bytes of data
written into selfie2.m

selfie1.m: exiting with exit code 0 and 1.05MB of mallocated memory

./selfie: this is selfie's mipster terminating selfie1.m with exit
code 0 and 1.16MB of mapped memory
```

compiling `selfie.c` *into a MIPSter executable* `selfie1.m`

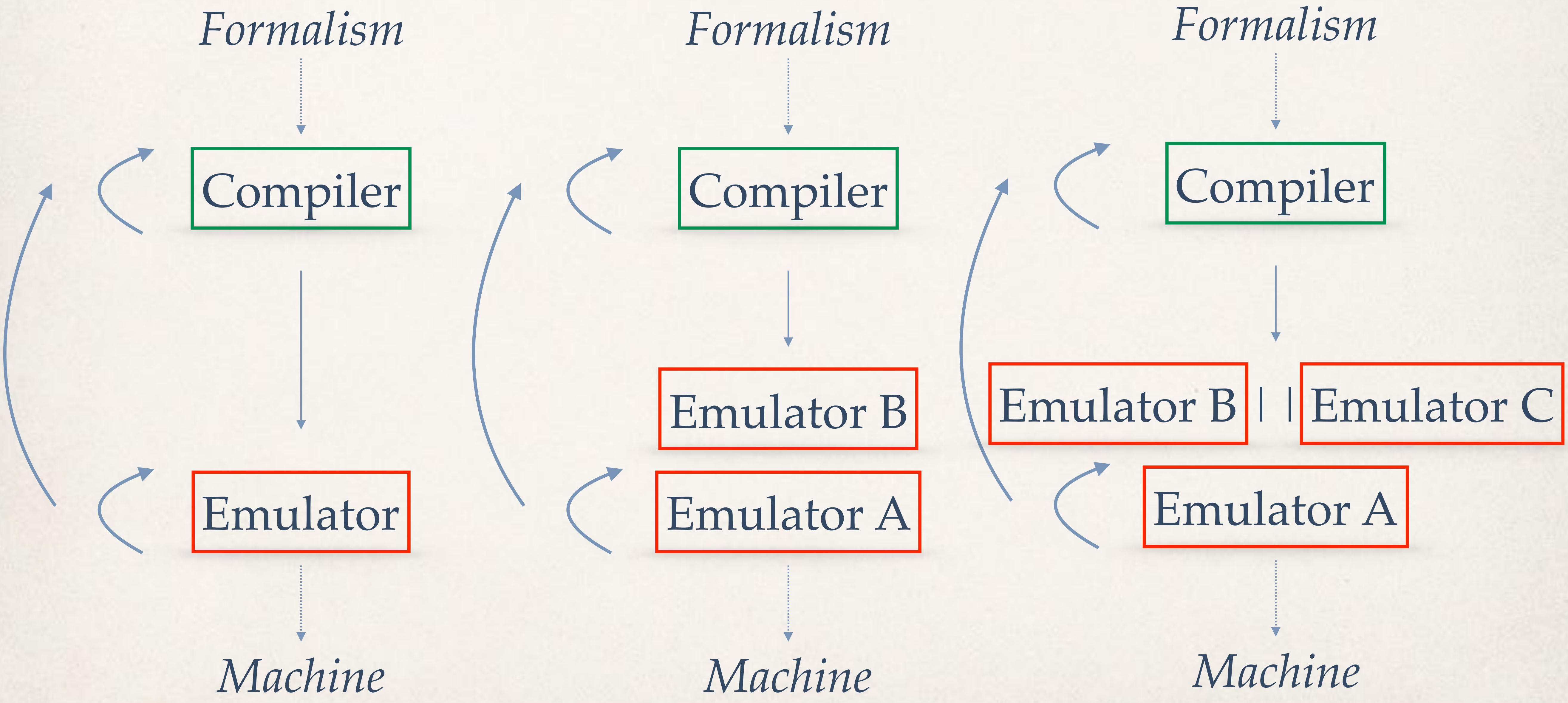
and

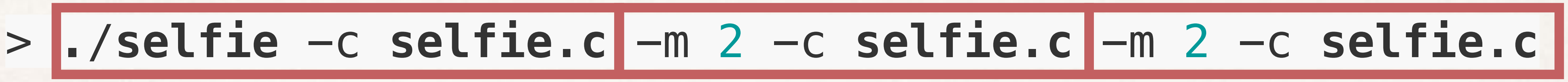
then running `selfie1.m` *to compile* `selfie.c`

into another MIPSter executable `selfie2.m`

*(**takes ~6 minutes**)*

Sandboxed Concurrency:
1-Week Homework Assignment





```
> ./selfie -c selfie.c -m 2 -c selfie.c -m 2 -c selfie.c
```

compiling `selfie.c` *with x86* `selfie` executable

and

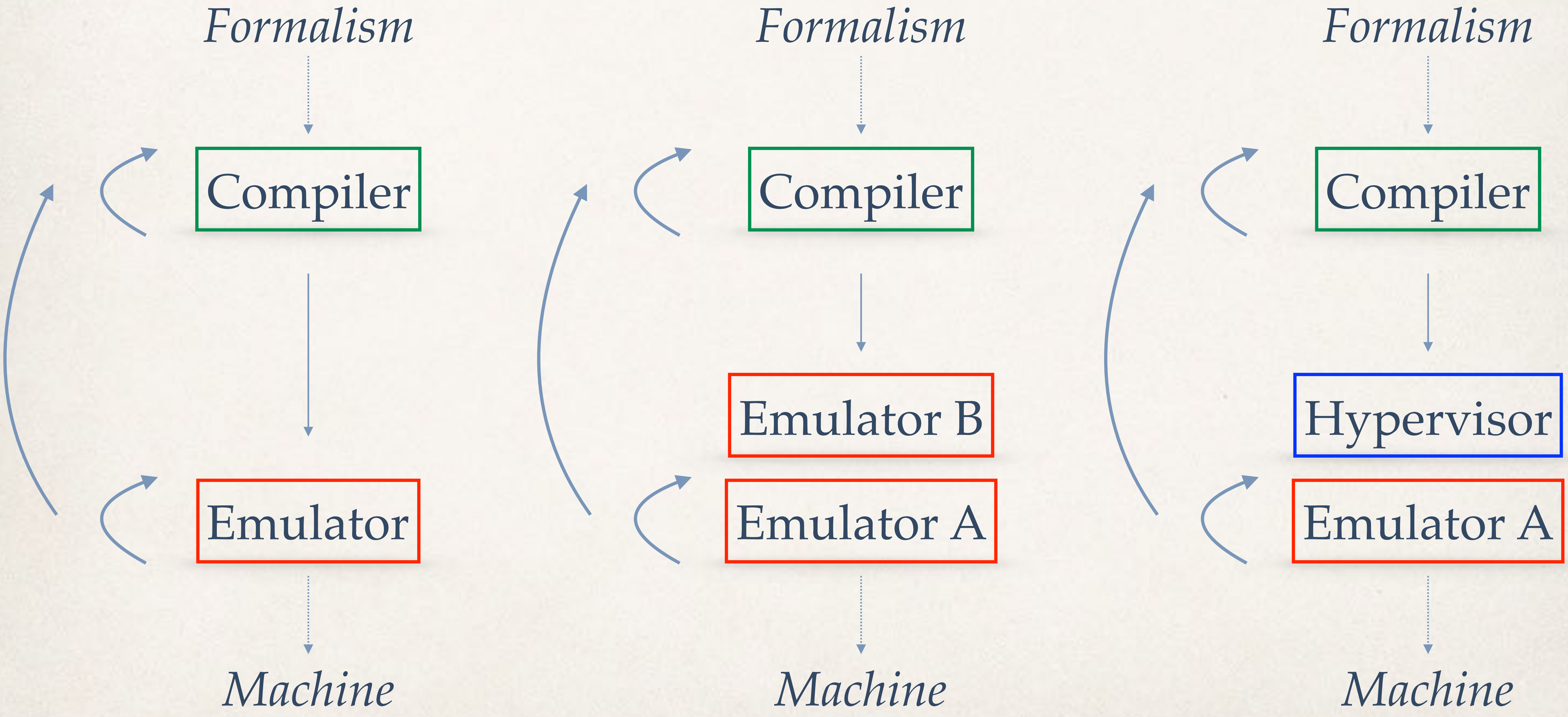
then running that executable to compile `selfie.c` *again*

and

then running that executable to compile `selfie.c` *again*

(takes ~24 hours)

Emulation versus Virtualization



```
> ./selfie -c selfie.c -m 2 -c selfie.c -y 2 -c selfie.c
```

compiling `selfie.c` *with x86* `selfie` *executable*

and

then running that executable to compile `selfie.c` *again*

and

then ***hosting*** *that executable in a virtual machine to compile* `selfie.c` *again*

(takes ~12 minutes)

Ongoing Work

Verification

- SAT/SMT Solvers (microsat/boolector)
- Symbolic Execution Engine (KLEE/SAGE)
- Inductive Theorem Prover (ACL2)

-> microsat in C* is as fast as in C (forget structs, arrays, &&, ||, goto)

ISAs

1. ELF binaries (taken from RISC-V port)
2. x86 support (how many instructions?)
3. ARM support? Any ARM people here?

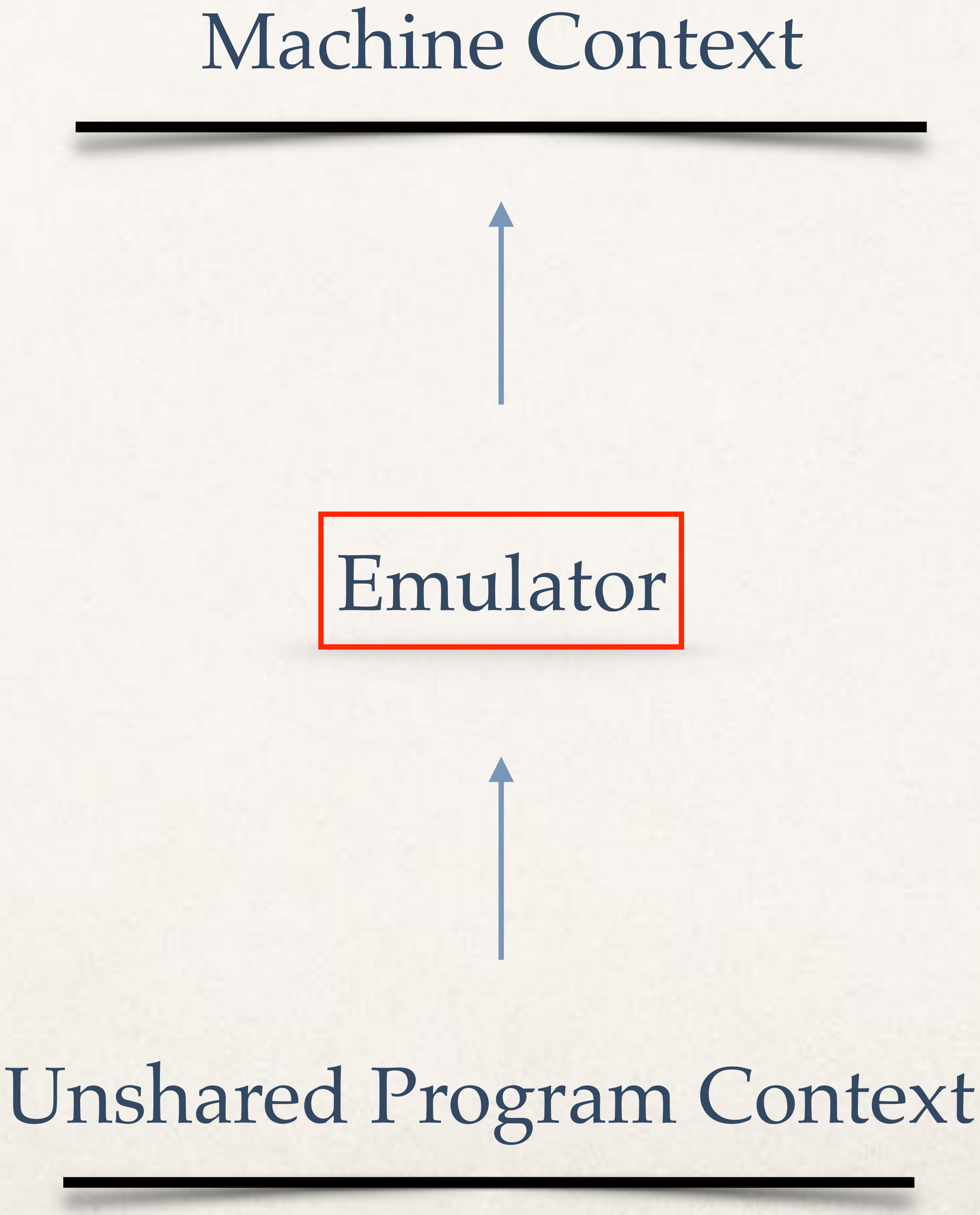
babysat this

```
./selfie -sat rivest.cnf
./selfie: this is selfie loading SAT instance rivest.cnf
./selfie: 7 clauses with 4 declared variables loaded from rivest.cnf
p cnf 4 7
2 3 -4 0
1 3 4 0
-1 2 4 0
-1 -2 3 0
-2 -3 4 0
-1 -3 -4 0
1 -2 -4 0
./selfie: rivest.cnf is satisfiable with -1 -2 3 4
```

What is the absolute simplest way of proving non-trivial properties of Selfie using Selfie, and what are these properties?

https://github.com/cksystemsteaching/selfie/tree/vipster

Emulation

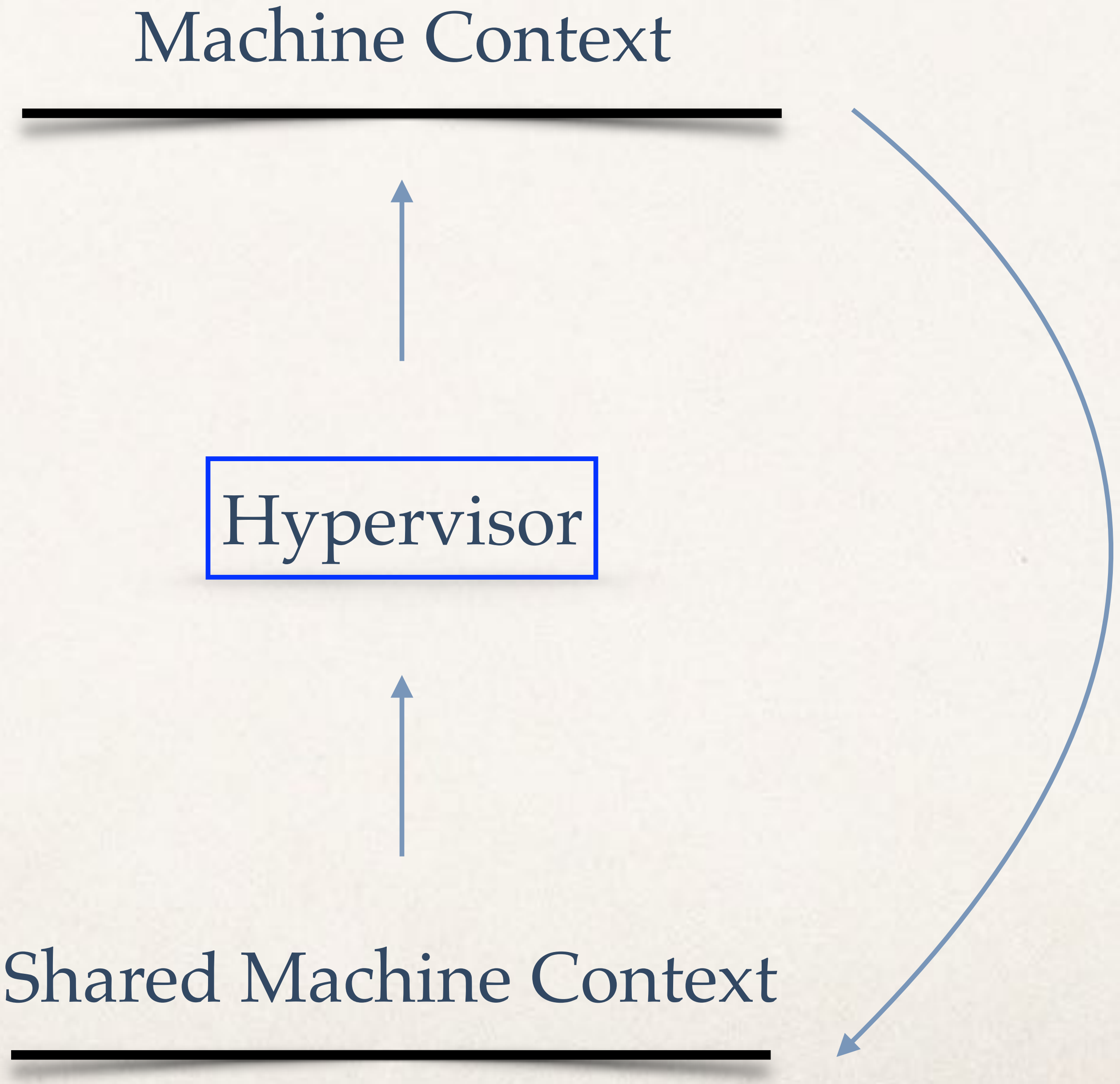


Machine Context

Emulator

Unshared Program Context

Virtualization



Machine Context

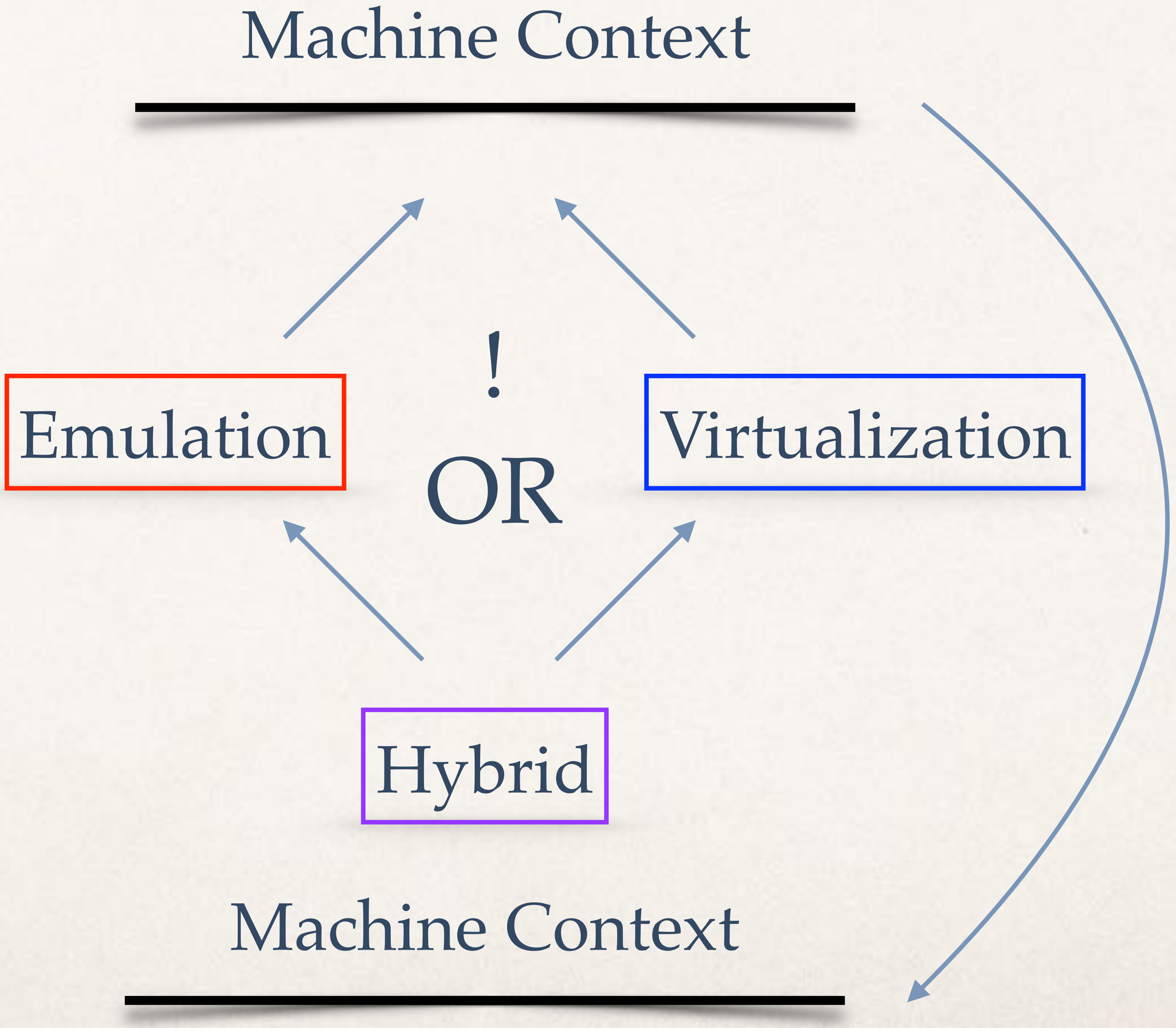
Hypervisor

Shared Machine Context

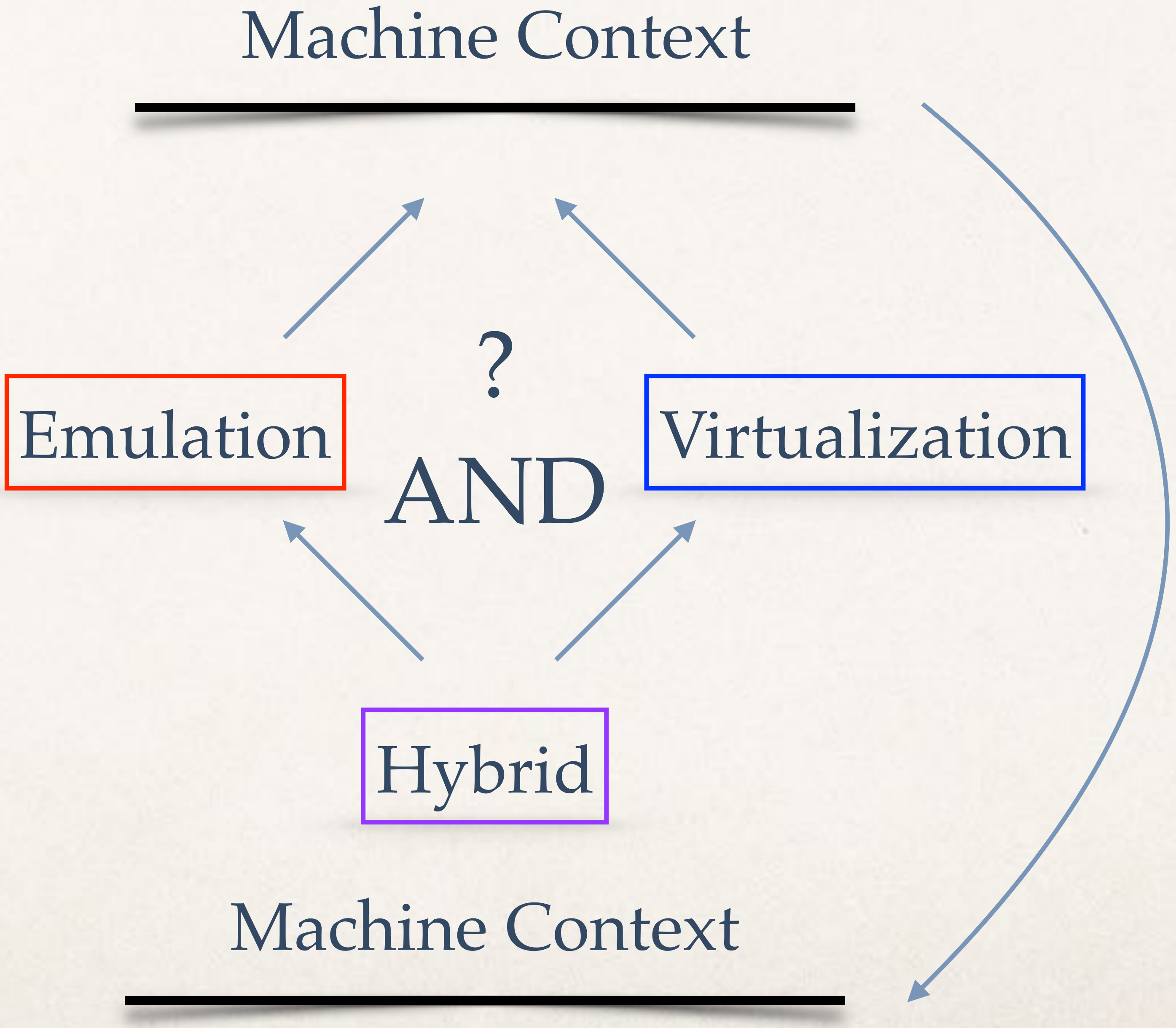
Proof Obligation

$$\frac{\text{Machine Context}}{\boxed{\text{Emulator}}} \overset{?}{=} \frac{\text{Machine Context}}{\boxed{\text{Hypervisor}}}$$

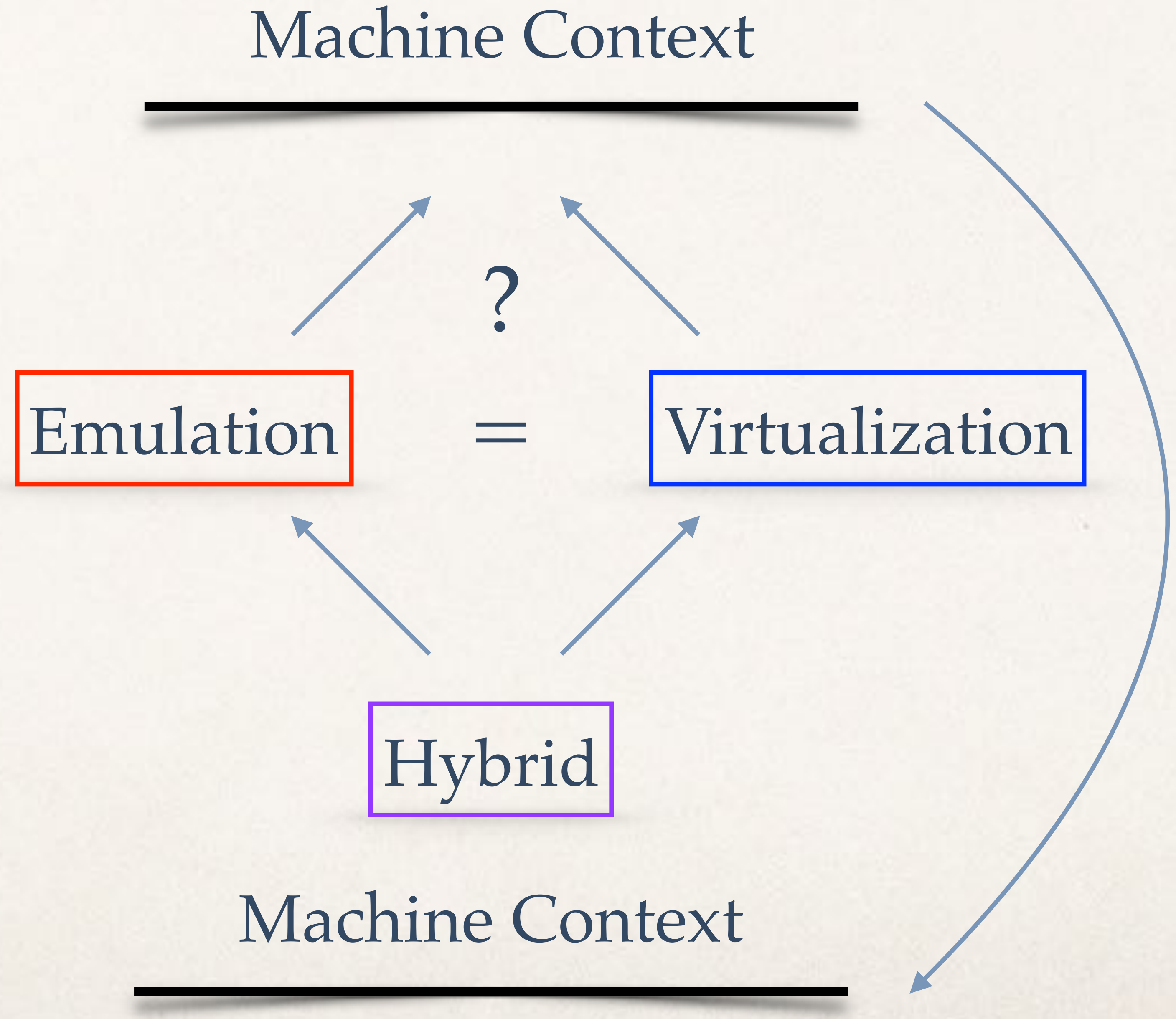
Mixter (T. Hütter, MS Thesis, 2017): Hybrid of Emulator & Hypervisor



Validation of Functional Equivalence?



Verification of Functional Equivalence?



Thank you!

acsd2018.cs.uni-salzburg.at