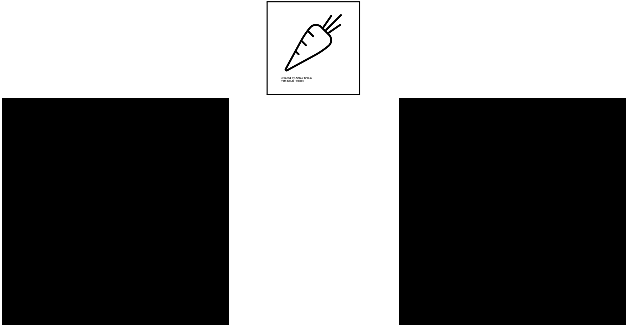
Tamarin: Concolic Disequivalence for MIPS

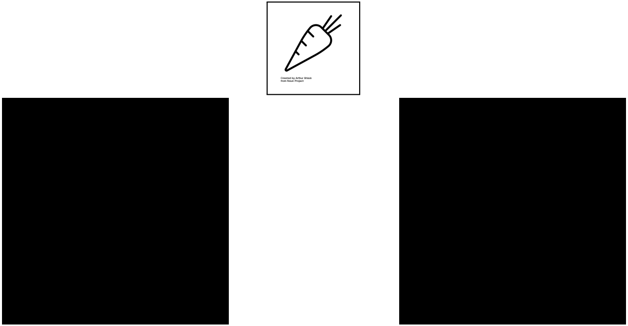
Abel Nieto

A Tale of Two Boxes

A Tale of Two Boxes

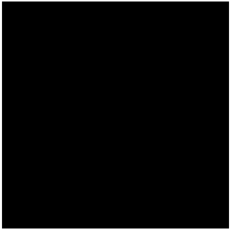


A Tale of Two Boxes

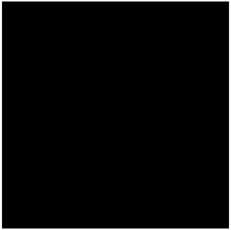


nom nom

A Tale of Two Boxes

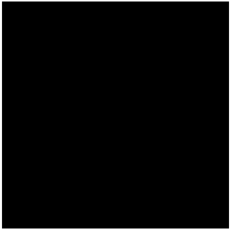


nom nom



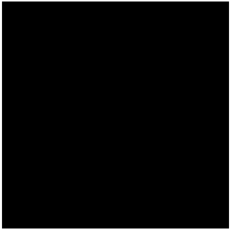
nom nom

A Tale of Two Boxes



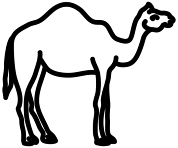
nom nom

=



nom nom

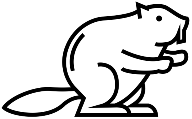
A Tale of Two Boxes



Created by H Alberto Gongora
from Noun Project

cmcm

$\neq$



Created by Iconic
from Noun Project

The Problem

Given MIPS program P_1 and P_2, when are they equivalent?

The Problem

Attempt 1: two programs are equivalent if they give the same output (resp.) for all inputs.

The Problem

Attempt 1: two programs are equivalent if they give the same output (resp.) for all inputs.

The Problem

Attempt 1: two programs are equivalent if they give the same output (resp.) for all inputs.

What's an input?

The Problem

Attempt 1: two programs are equivalent if they give the same output (resp.) for all inputs.

What's an input? Register \$1 and \$2.

The Problem

Attempt 1: two programs are equivalent if they give the same output (resp.) for all inputs.

What's an input? Register \$1 and \$2.
What's an output?

The Problem

Attempt 1: two programs are equivalent if they give the same output (resp.) for all inputs.

What's an input? Register \$1 and \$2.
What's an output? Register \$3.

The Problem

Attempt 1: two programs are equivalent if they give the same output (resp.) for all inputs.

What's an input? Register \$1 and \$2.
What's an output? Register \$3.

Don't care about (most) CPU interrupts/IO.

The Problem

Attempt 1: two programs are equivalent if they give the same output (resp.) for all inputs.

The Problem

Attempt 1: two programs are equivalent if they give the same output (resp.) for all inputs.

Problem: undecidable via Rice's theorem.

The Problem

Attempt 2: two programs are S-equivalent if they cannot be told apart after S steps.

The Problem

Attempt 2: two programs are S-equivalent if they cannot be told apart after S steps.

S-equivalent (e.g. for $S = 10$), but not equivalent:

```
# P_1
add $3, $1, $2
```

```
# P_2
  add $4, $0, 1  # counter
  add $5, $0, 42 # upper bound
loop:
  slt $6, $4, $5
  beq $6, $0, end
  add $4, $4, 1
  beq $0, $0, loop
end:
  add $3, $1, $1
```

The Problem

Attempt 2: two programs are S-equivalent if they **cannot be told apart** after S steps.

R_1	R_2	S-equiv

The Problem

Attempt 2: two programs are S-equivalent if they **cannot be told apart** after S steps.

R_1	R_2	S-equiv
v	v	yes

The Problem

Attempt 2: two programs are S-equivalent if they **cannot be told apart** after S steps.

R_1	R_2	S-equiv
v	v	yes
v	$w \neq v$	no

The Problem

Attempt 2: two programs are S-equivalent if they **cannot be told apart** after S steps.

R_1	R_2	S-equiv
v	v	yes
v	$w \neq v$	no
v	error	no

The Problem

Attempt 2: two programs are S-equivalent if they **cannot be told apart** after S steps.

R_1	R_2	S-equiv
v	v	yes
v	$w \neq v$	no
v	error	no
error	error	yes

The Problem

Attempt 2: two programs are S-equivalent if they **cannot be told apart** after S steps.

R_1	R_2	S-equiv
v	v	yes
v	$w \neq v$	no
v	error	no
error	error	yes
non-termination	???	yes
...		

The Problem

Attempt 2: two programs are S-equivalent if they **cannot be told apart** after S steps.

Which inputs?

The Problem

Attempt 2: two programs are S-equivalent if they **cannot be told apart** after S steps.

Which inputs?

Try some inputs by hand: low coverage, fast (unit tests)

The Problem

Attempt 2: two programs are S-equivalent if they **cannot be told apart** after S steps.

Which inputs?

Try some inputs by hand: low coverage, fast (unit tests)

Try all 2^{64}values of \$1 and \$2: high coverage, slow (but decidable)

The Problem

Attempt 2: two programs are S-equivalent if they **cannot be told apart** after S steps.

Which inputs?

Try some inputs by hand: low coverage, fast (unit tests)

Try all 2^{64}values of \$1 and \$2: high coverage, slow (but decidable)

Tamarin: use concolic testing to check if the programs are disequivalent