A system for testing specifications of CPU semantics

or,
What I did on my summer vacation

Lindsey Kuper

Static analysis of executables: what, why, and how?

Static analysis of executables: what, why, and how?

- Abstract interpretation: a *sound overapproximation* of the behavior of a concrete system.

Static analysis of executables: what, why, and how?

- Abstract interpretation: a *sound overapproximation* of the behavior of a concrete system.
- "WYSINWYX: What You See Is Not What You Execute" (Balakrishnan *et al.* 2005)

Static analysis of executables: what, why, and how?

- Abstract interpretation: a *sound overapproximation* of the behavior of a concrete system.
- "WYSINWYX: What You See Is Not What You Execute" (Balakrishnan *et al.* 2005)
- What kinds of static analysis can one do on executables?

Static analysis of executables: what, why, and how?

- Abstract interpretation: a *sound overapproximation* of the behavior of a concrete system.
- "WYSINWYX: What You See Is Not What You Execute" (Balakrishnan *et al.* 2005)
- What kinds of static analysis can one do on executables?
  - Value-set analysis (VSA)

Static analysis of executables: what, why, and how?

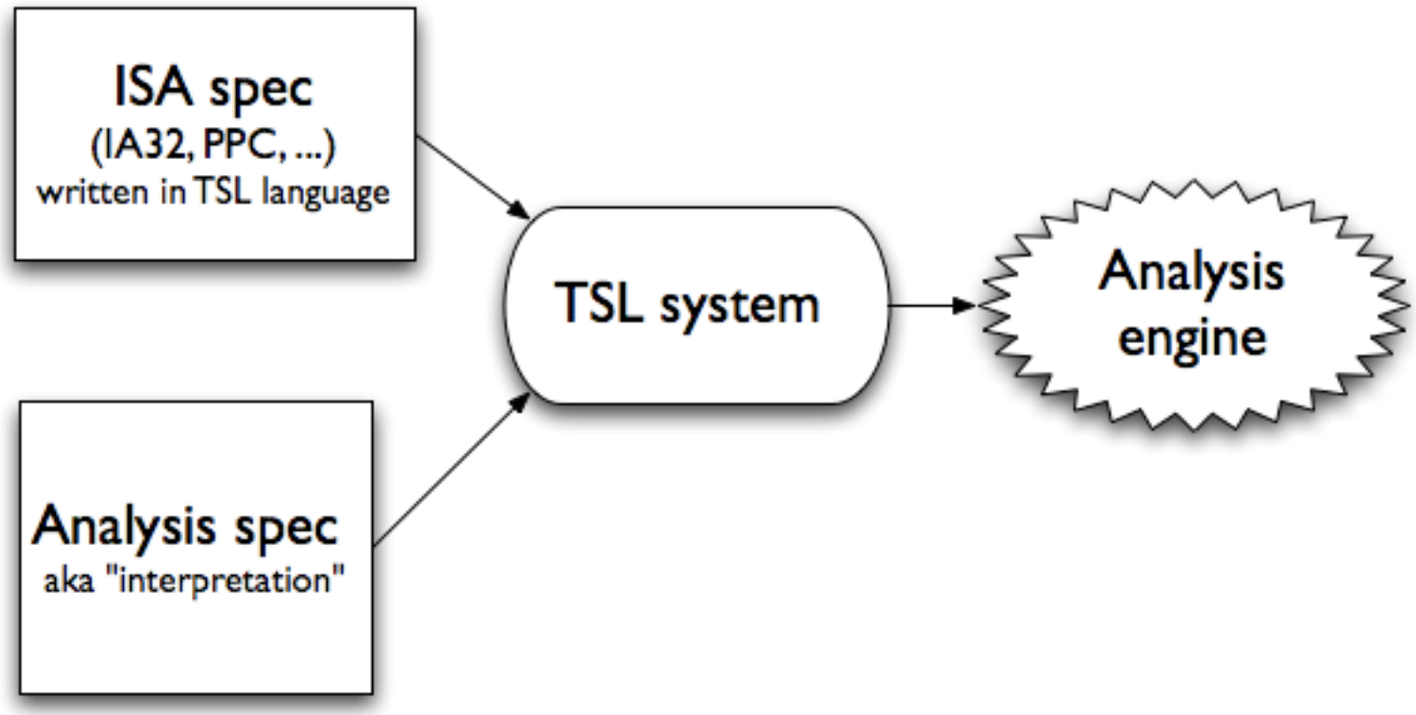
- Abstract interpretation: a *sound overapproximation* of the behavior of a concrete system.
- "WYSINWYX: What You See Is Not What You Execute" (Balakrishnan *et al.* 2005)
- What kinds of static analysis can one do on executables?
  - Value-set analysis (VSA)
  - Quantifier-free bit-vector (QFBV) analysis

Static analysis of executables: what, why, and how?

- Abstract interpretation: a *sound overapproximation* of the behavior of a concrete system.
- "WYSINWYX: What You See Is Not What You Execute" (Balakrishnan *et al.* 2005)
- What kinds of static analysis can one do on executables?
  - Value-set analysis (VSA)
  - Quantifier-free bit-vector (QFBV) analysis
  - ...

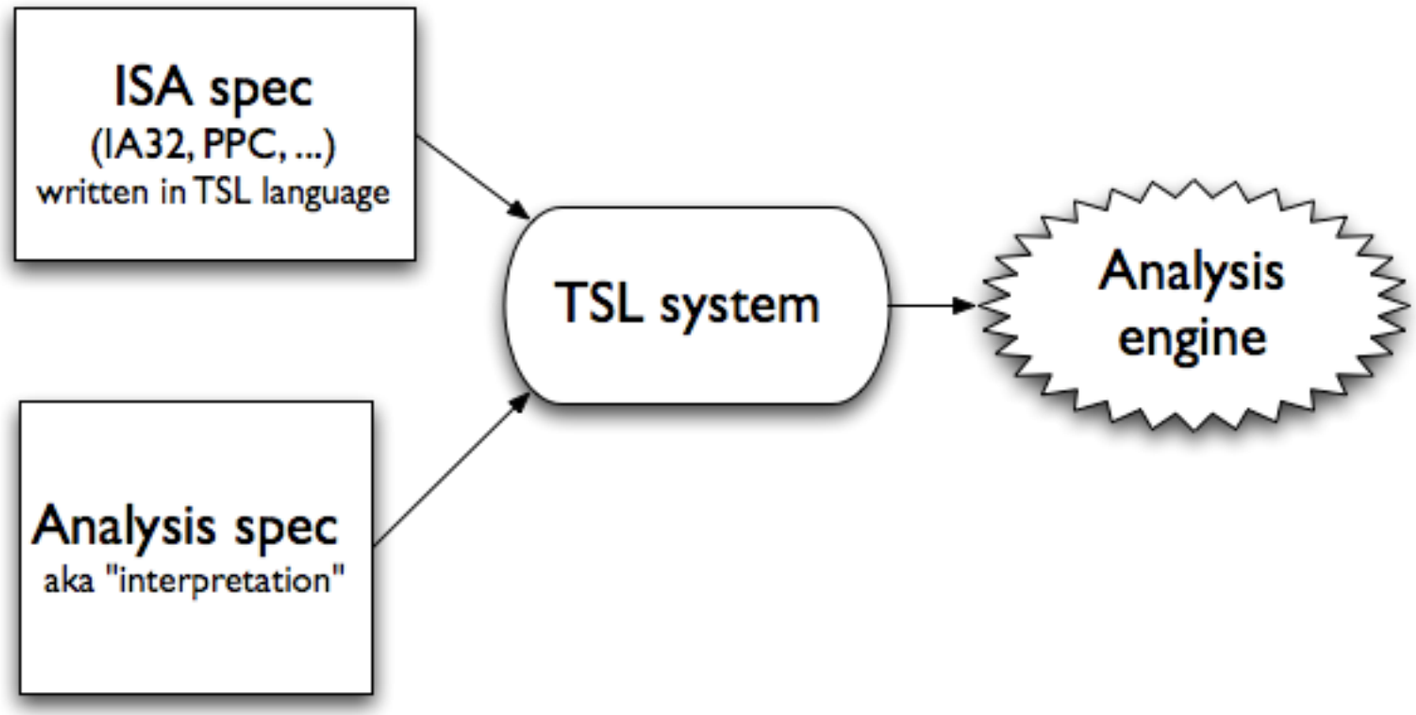
The TSL testing problem

The TSL testing problem



Lim, J, and Reps, T. , "A System for Generating Static Analyzers from Machine Instructions", CC '08

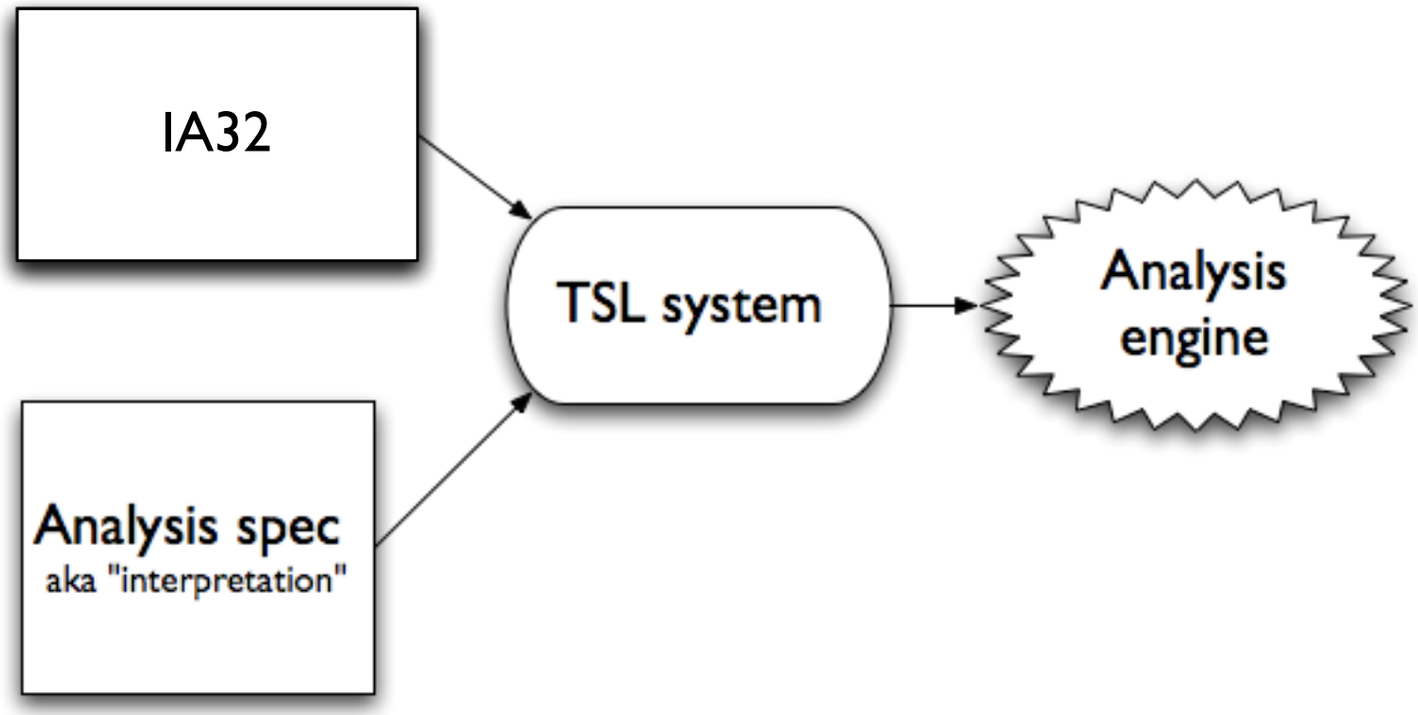
The TSL testing problem



- TSL (Transformer Specification Language) lets us *generate static analyzers from specifications*. Great!

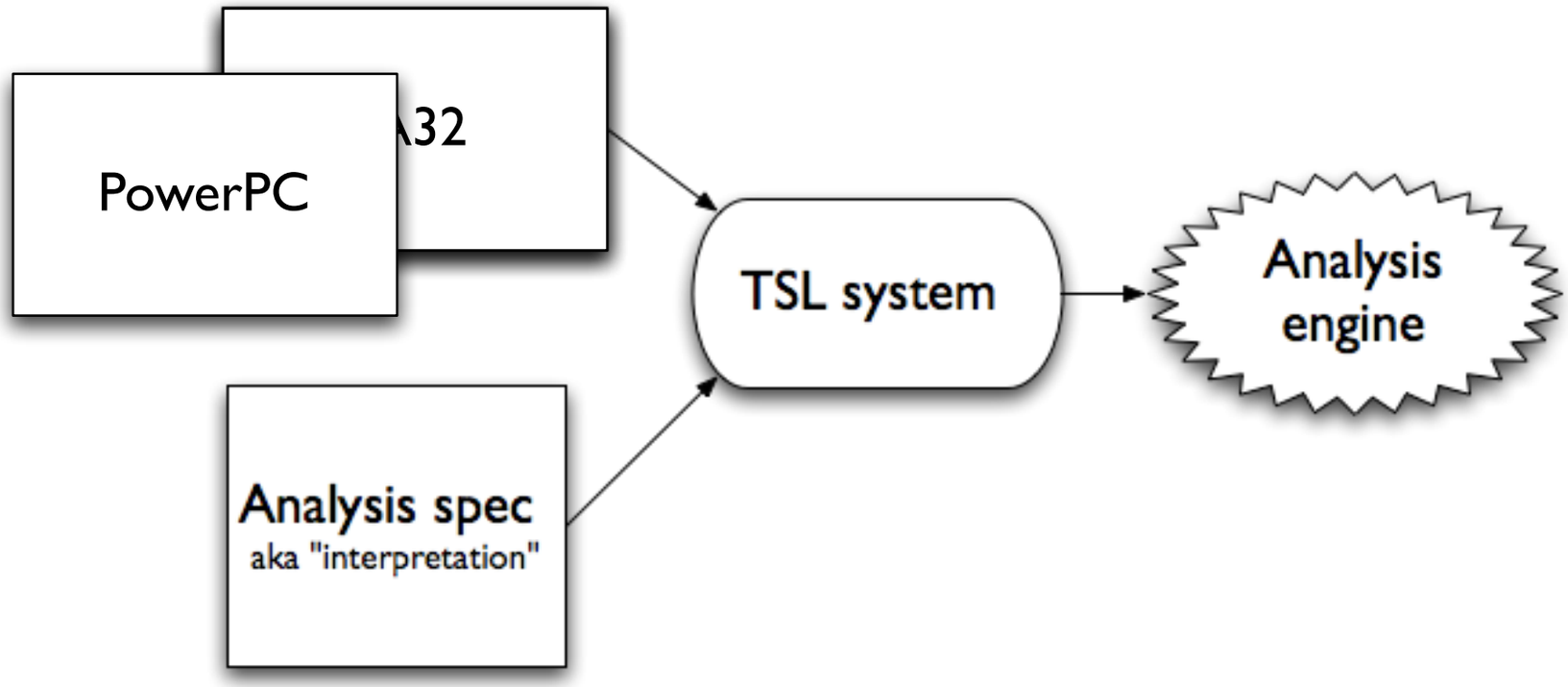
Lim, J, and Reps, T. , "A System for Generating Static Analyzers from Machine Instructions", CC '08

The TSL testing problem



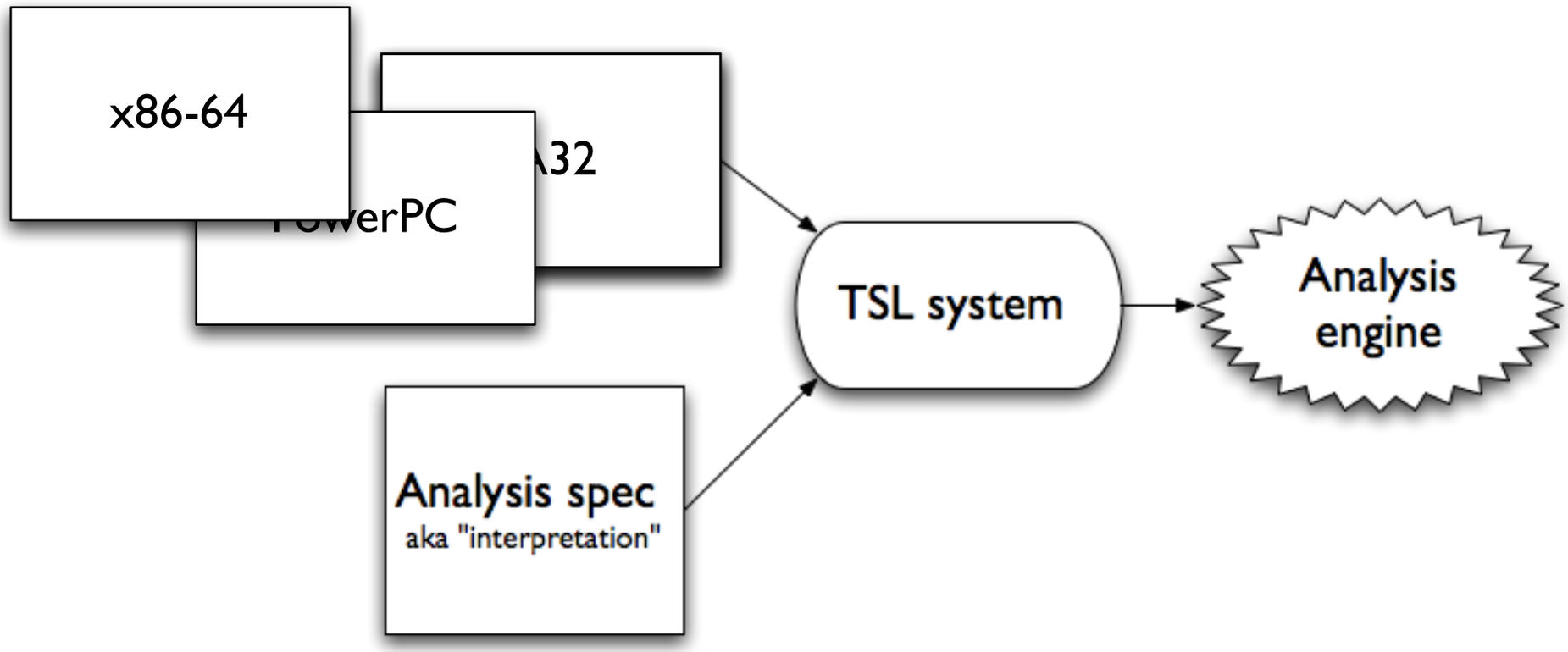
- TSL (Transformer Specification Language) lets us *generate static analyzers from specifications*. Great!

The TSL testing problem



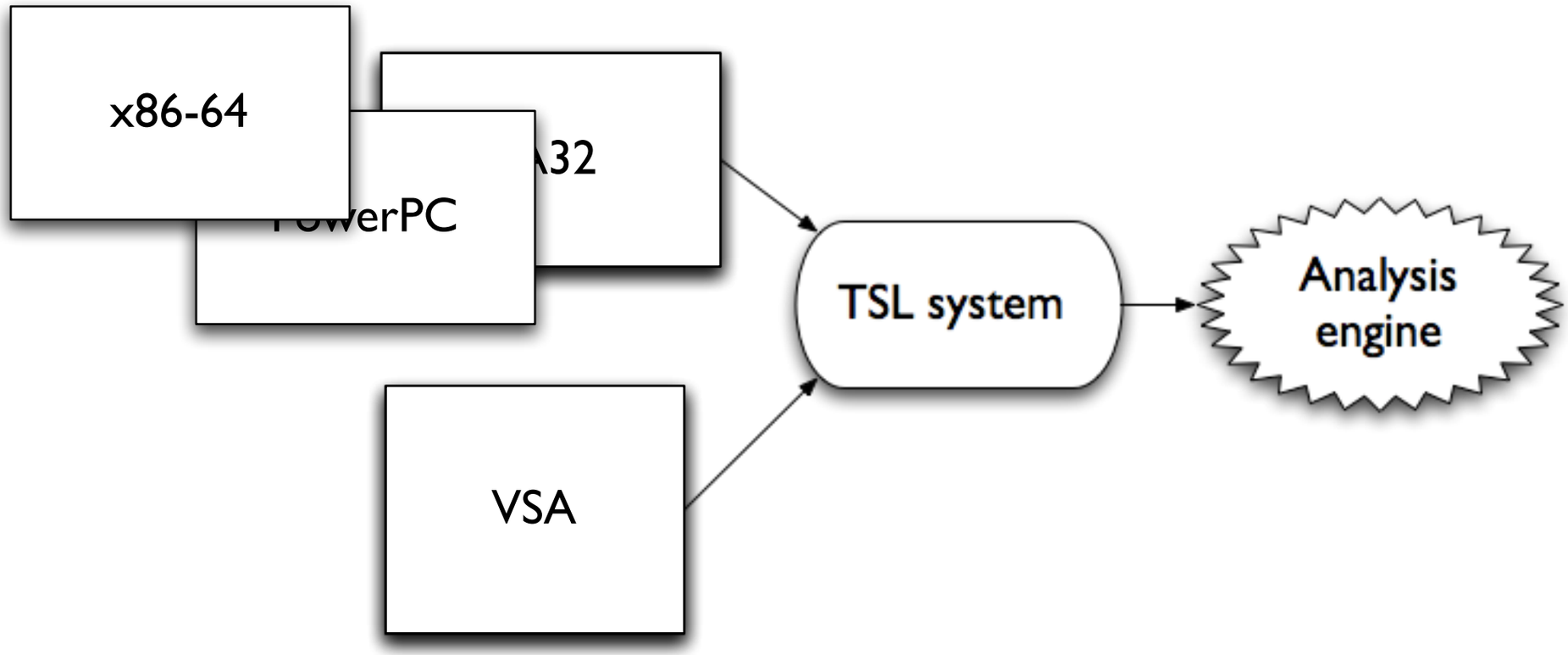
- TSL (Transformer Specification Language) lets us *generate static analyzers from specifications*. Great!

The TSL testing problem



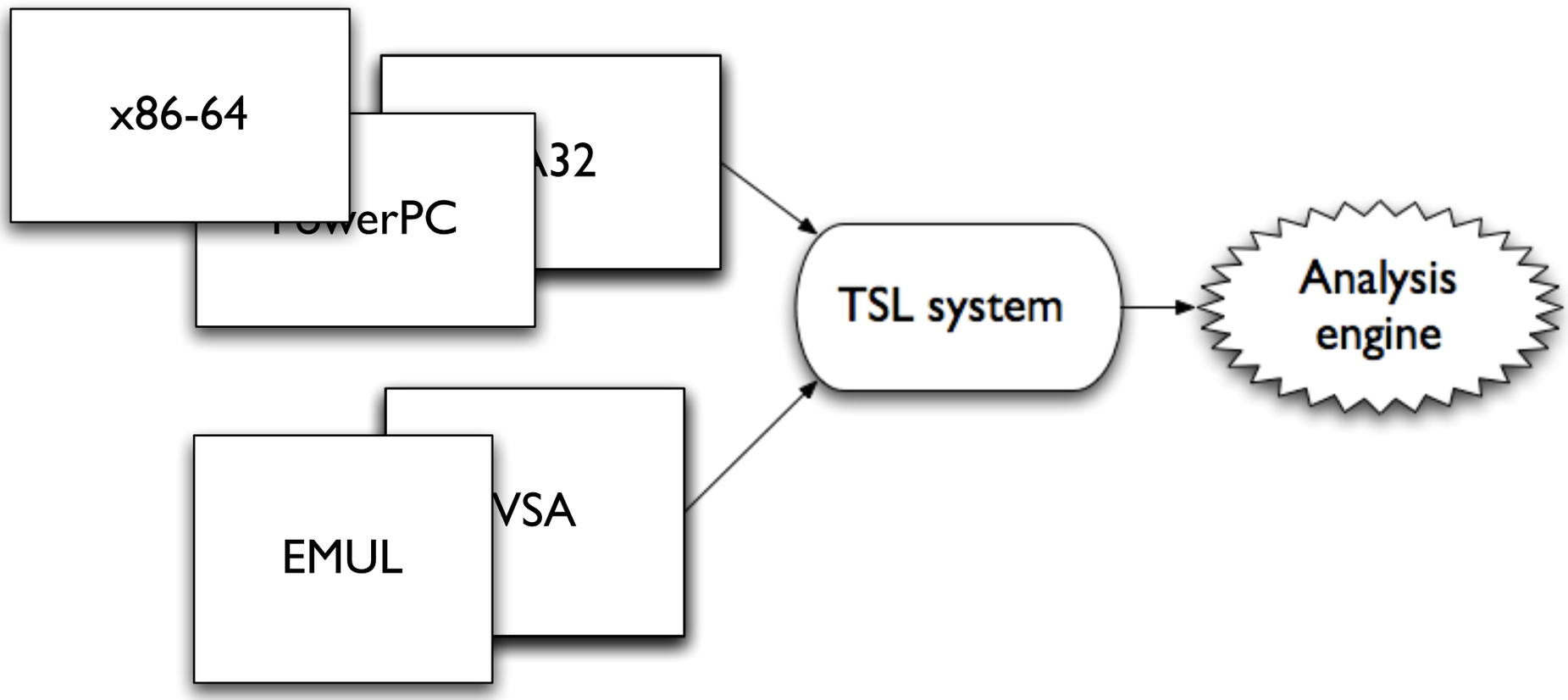
- TSL (Transformer Specification Language) lets us *generate static analyzers from specifications*. Great!

The TSL testing problem



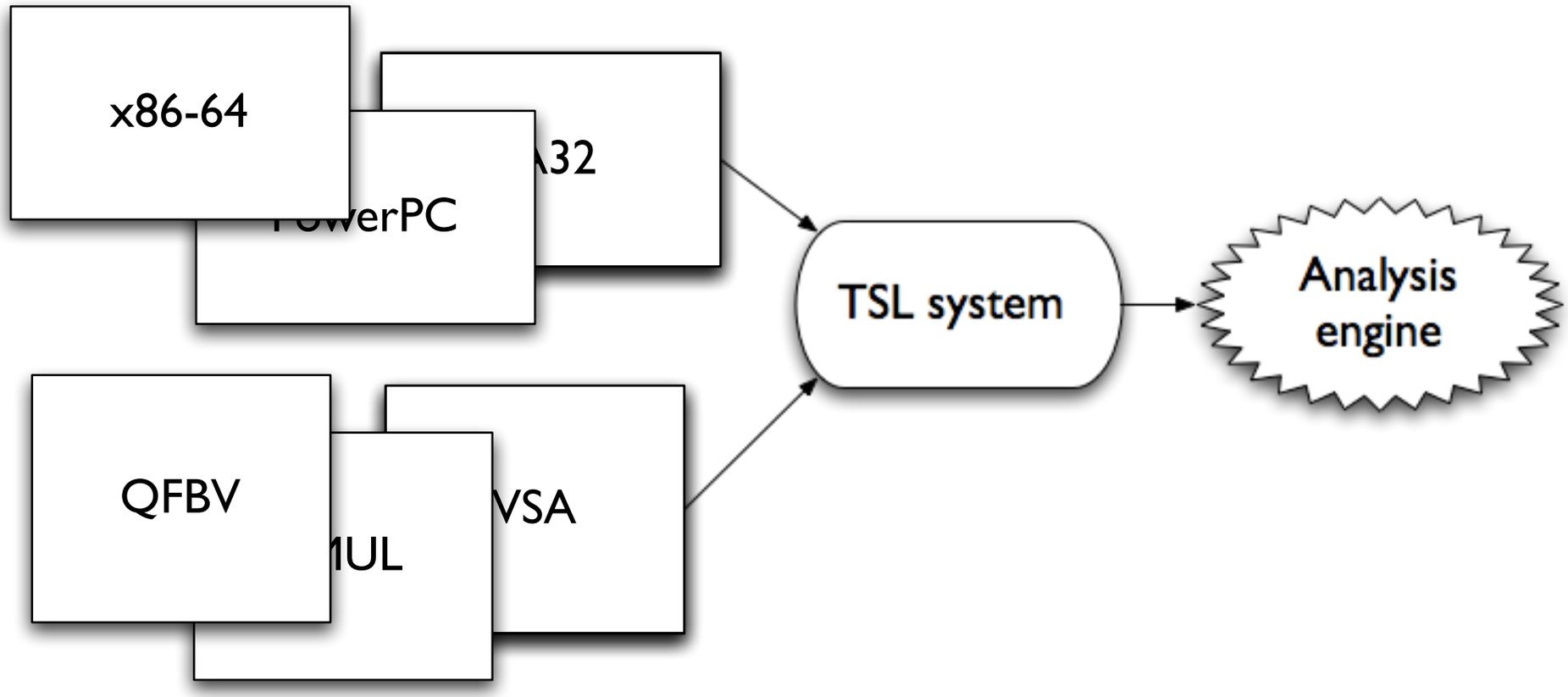
- TSL (Transformer Specification Language) lets us *generate static analyzers from specifications*. Great!

The TSL testing problem



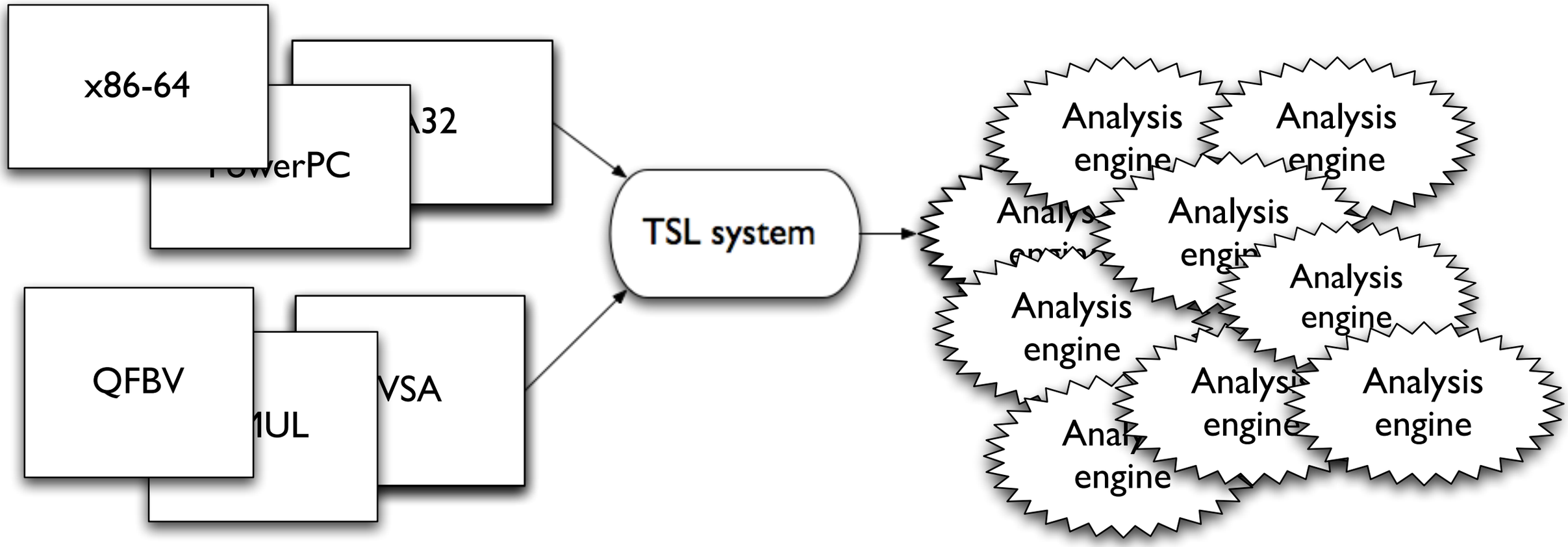
- TSL (Transformer Specification Language) lets us *generate static analyzers from specifications*. Great!

The TSL testing problem



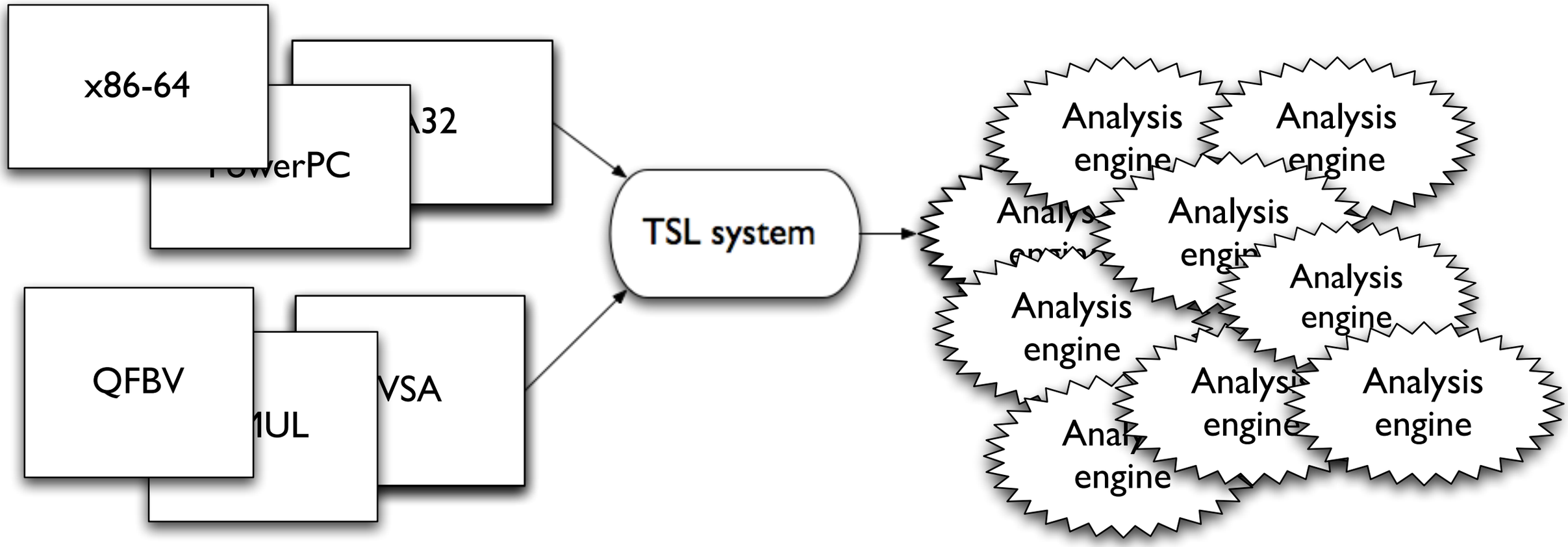
- TSL (Transformer Specification Language) lets us *generate static analyzers from specifications*. Great!

The TSL testing problem



- TSL (Transformer Specification Language) lets us *generate static analyzers from specifications*. Great!

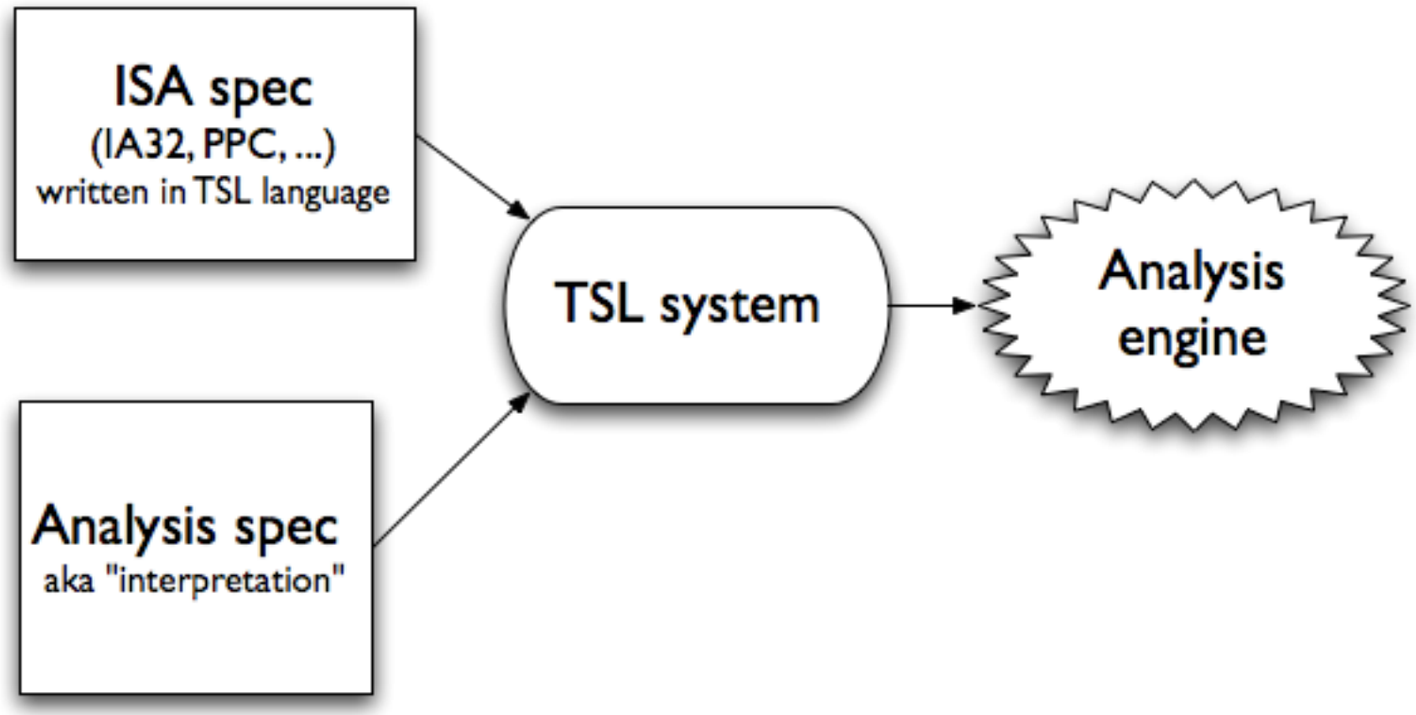
The TSL testing problem



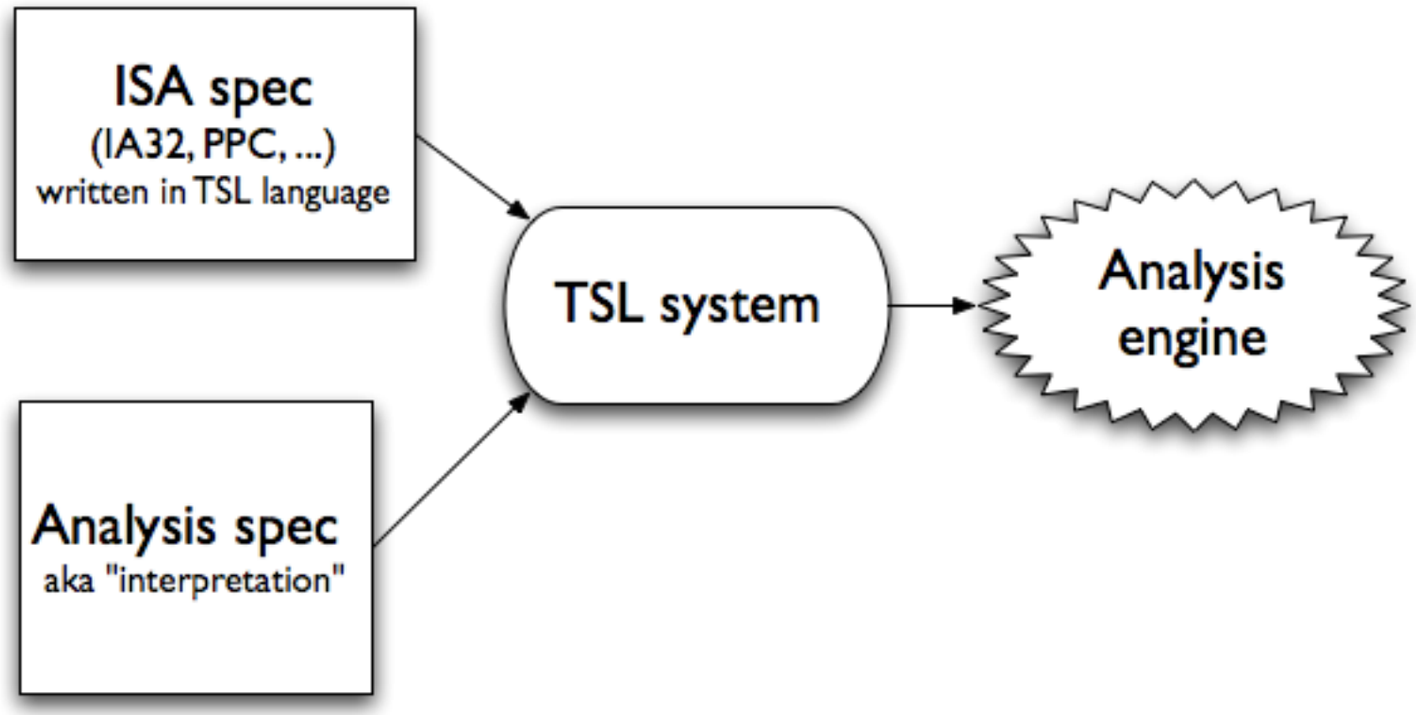
- TSL (Transformer Specification Language) lets us *generate static analyzers from specifications*. Great!
- But how do we know if the generated analysis engines (*multiplicatively many!*) are **correct**?

Our approach

Our approach

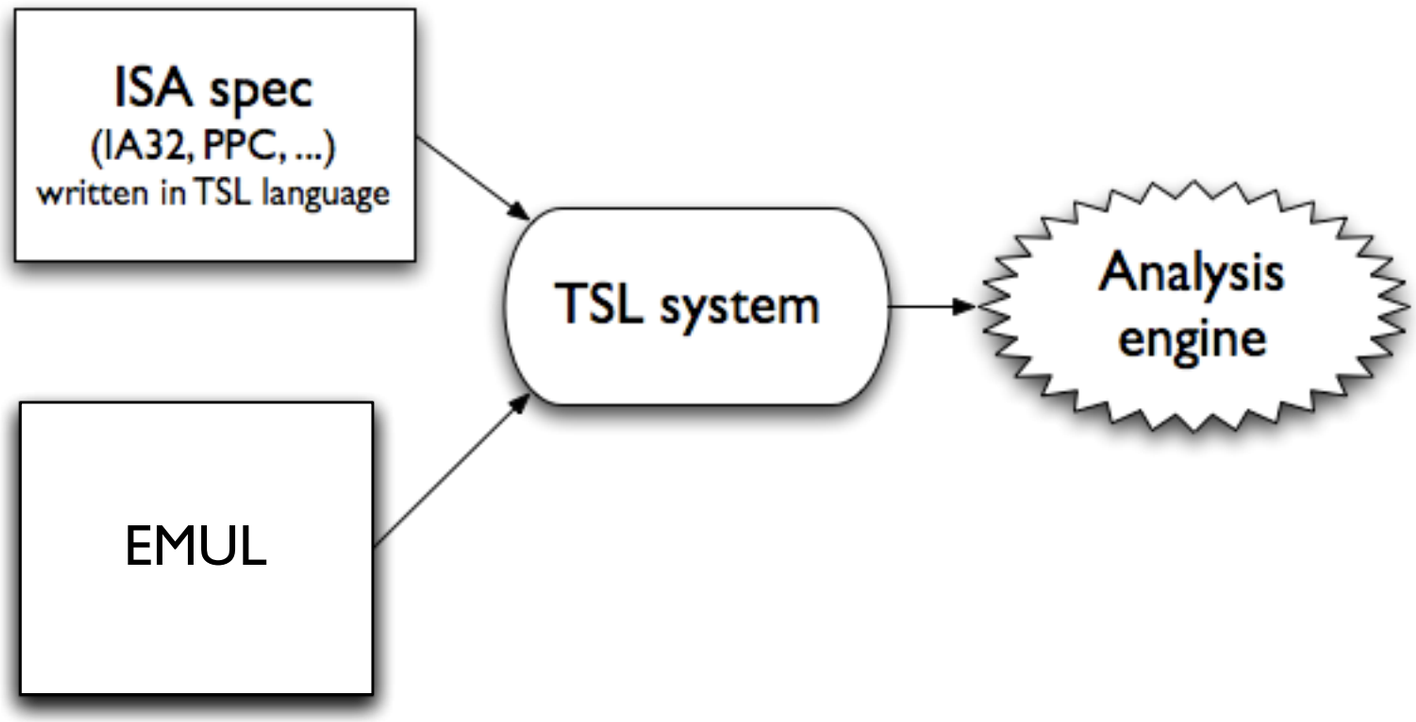


Our approach



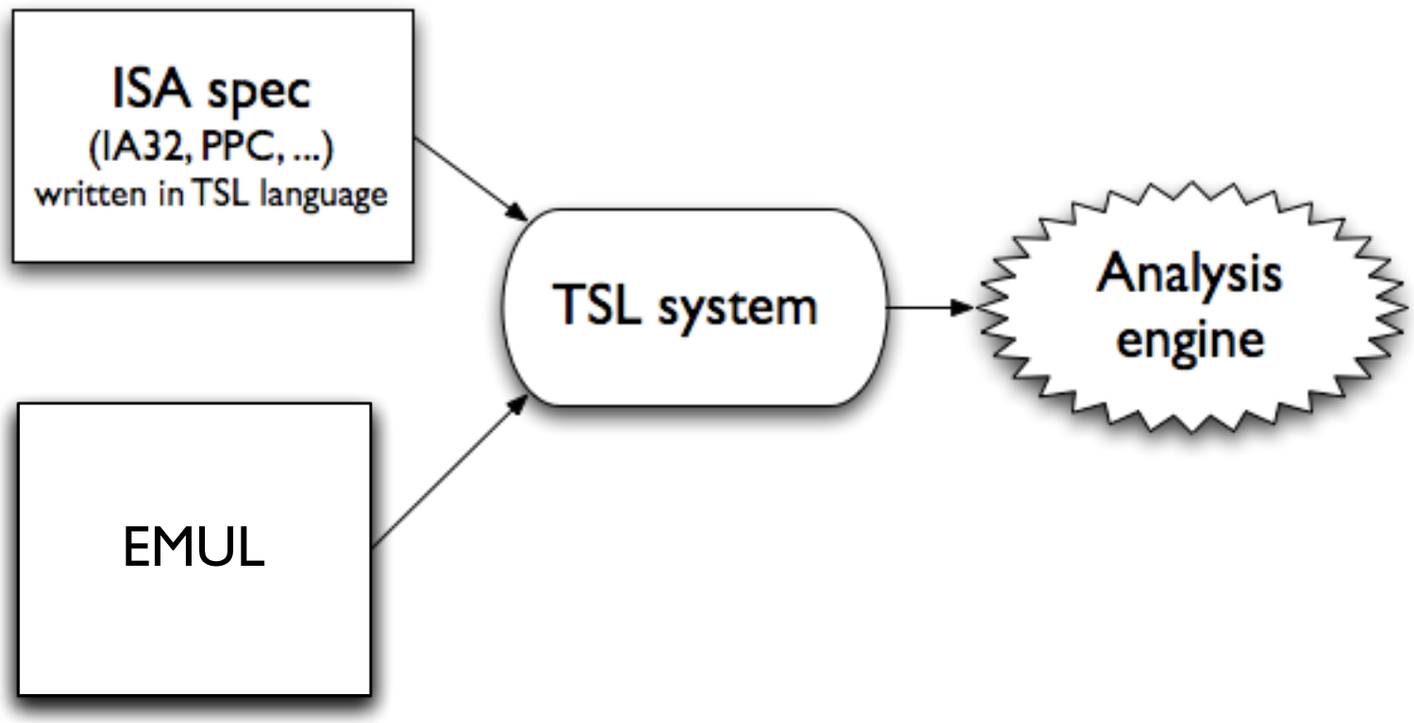
- Narrow the focus to testing just the ISA specs.

Our approach



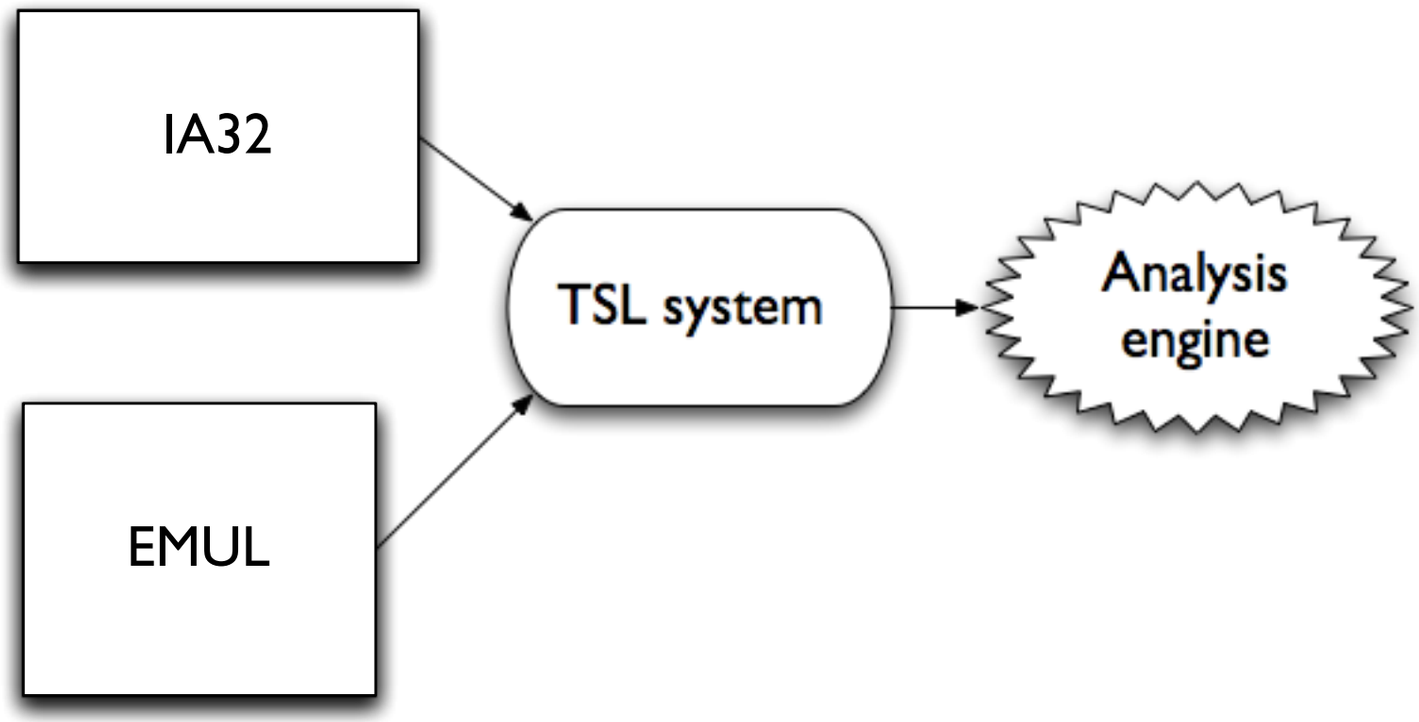
- Narrow the focus to testing just the ISA specs.

Our approach



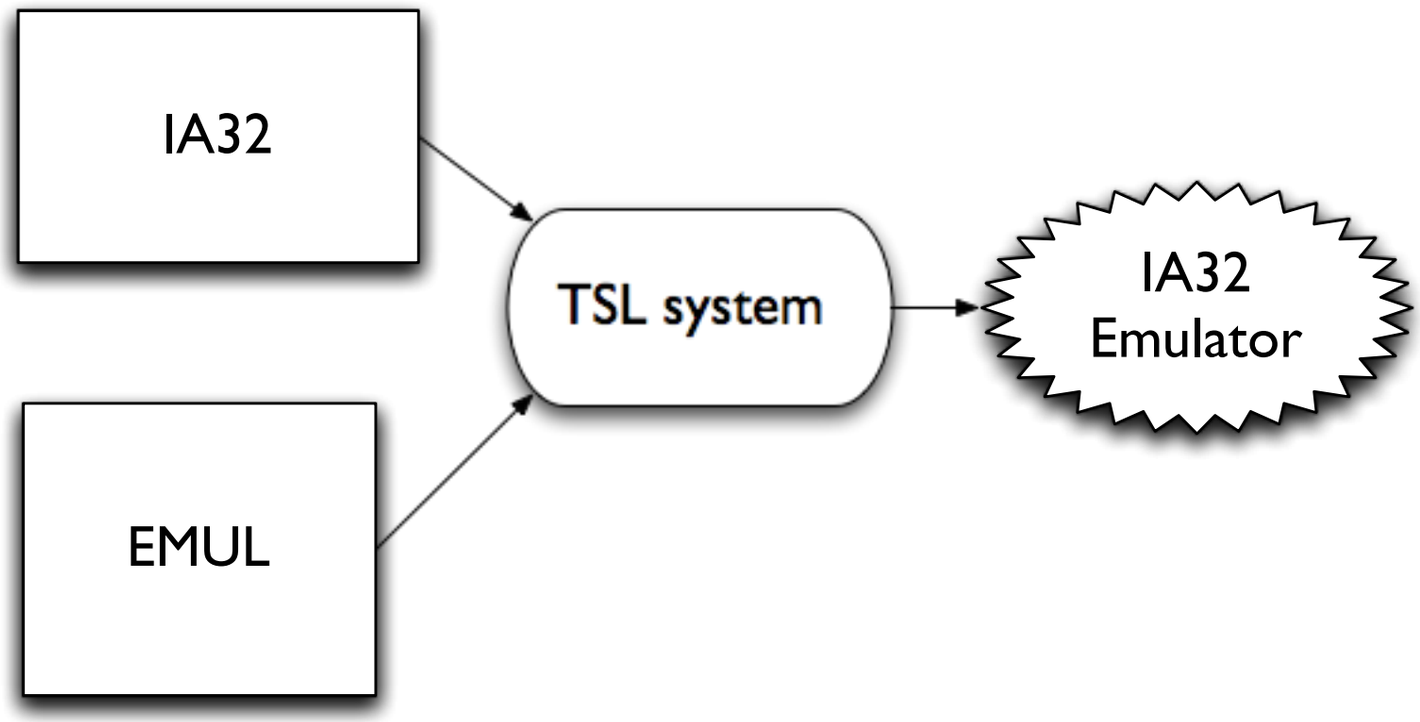
- Narrow the focus to testing just the ISA specs.
- Can we really *isolate* an ISA spec? We can come close by using EMUL, the "simplest" interpretation.

Our approach



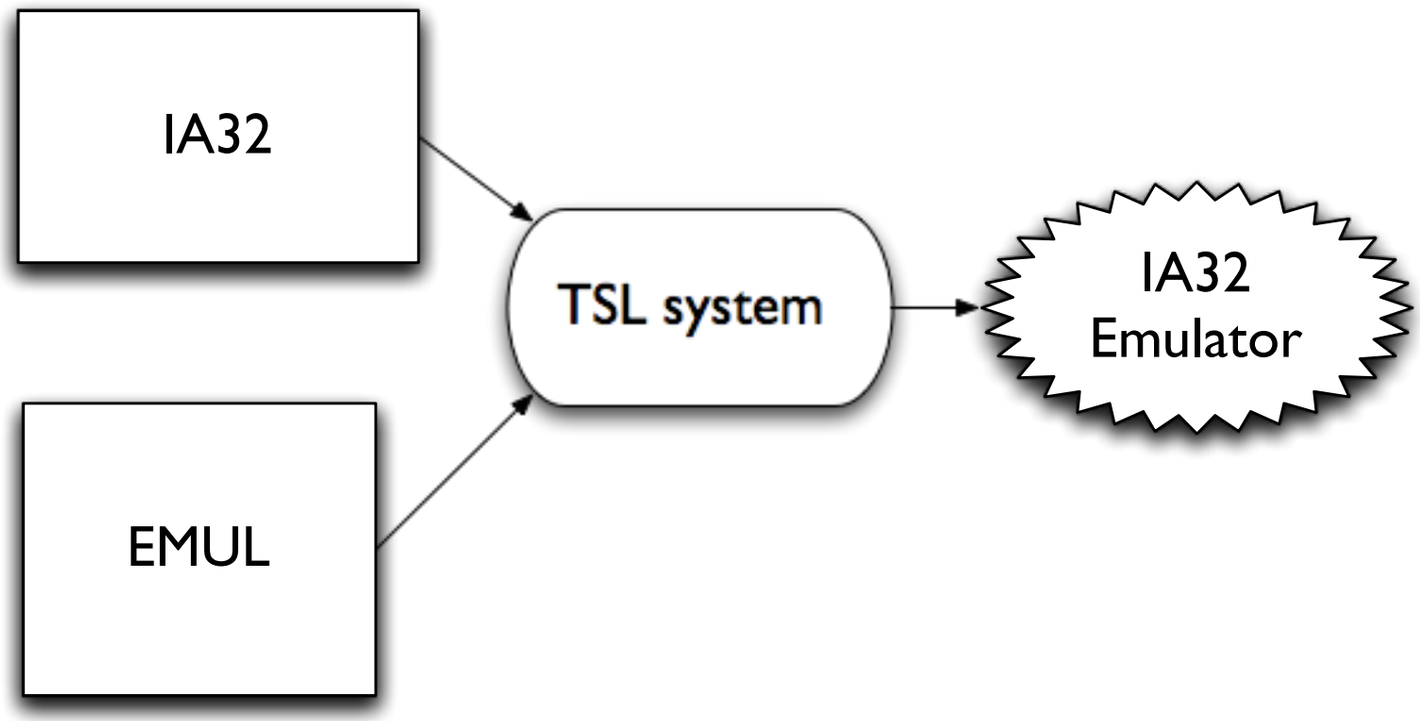
- Narrow the focus to testing just the ISA specs.
- Can we really *isolate* an ISA spec? We can come close by using EMUL, the "simplest" interpretation.

Our approach



- Narrow the focus to testing just the ISA specs.
- Can we really *isolate* an ISA spec? We can come close by using EMUL, the "simplest" interpretation.

Our approach



- Narrow the focus to testing just the ISA specs.
- Can we really *isolate* an ISA spec? We can come close by using EMUL, the "simplest" interpretation.
- And for now, start with IA32.

Goal for the summer

Goal for the summer

- **Find out how complete and precise our IA32 TSL specification is...**

Goal for the summer

- **Find out how complete and precise our IA32 TSL specification is...**
  - ...by generating an IA32 emulator, then comparing the emulator to the real processor.

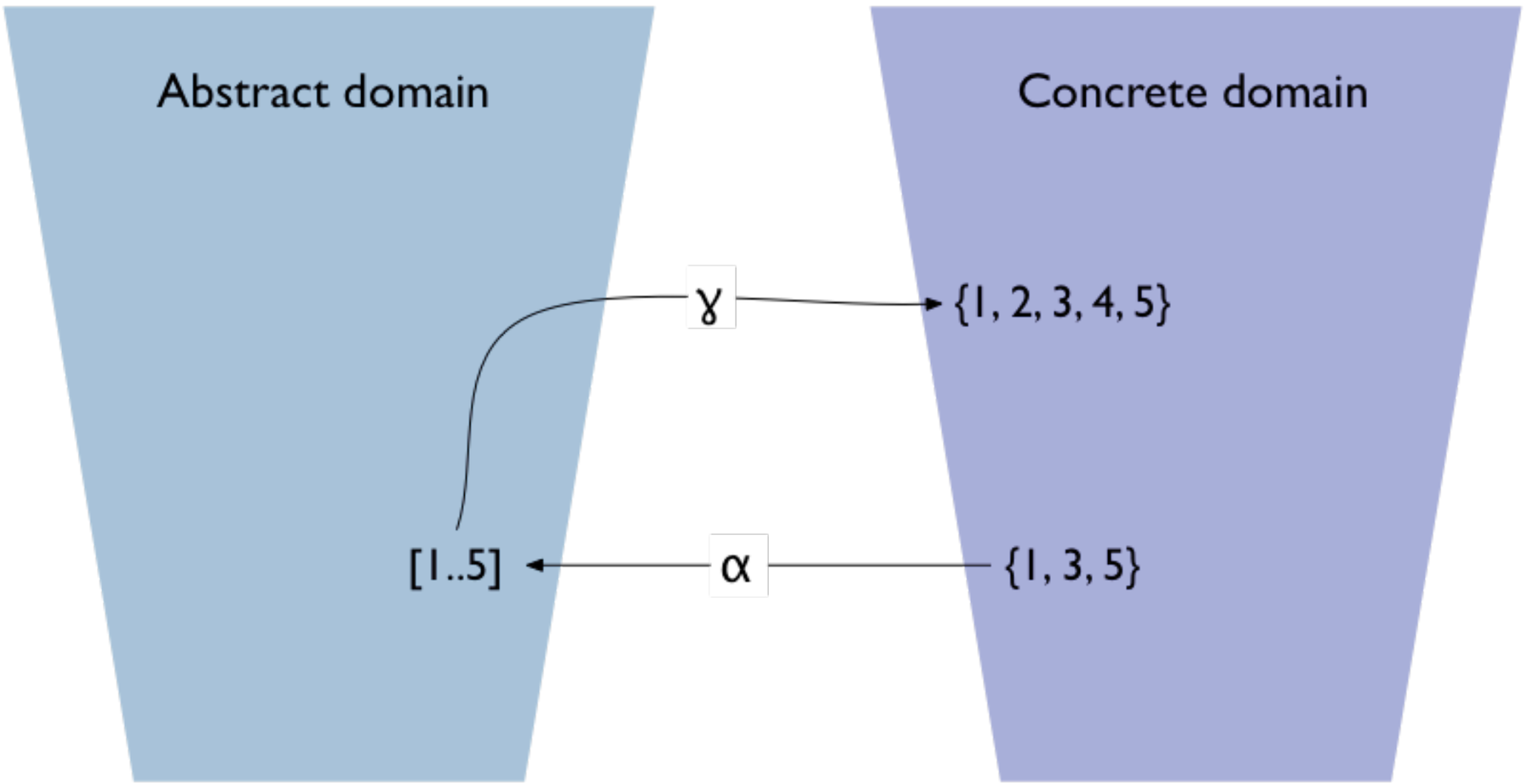
Goal for the summer

- **Find out how complete and precise our IA32 TSL specification is...**
  - ...by generating an IA32 emulator, then comparing the emulator to the real processor.
  - If resulting states differ on the same inputs, the spec was (*probably*) buggy.

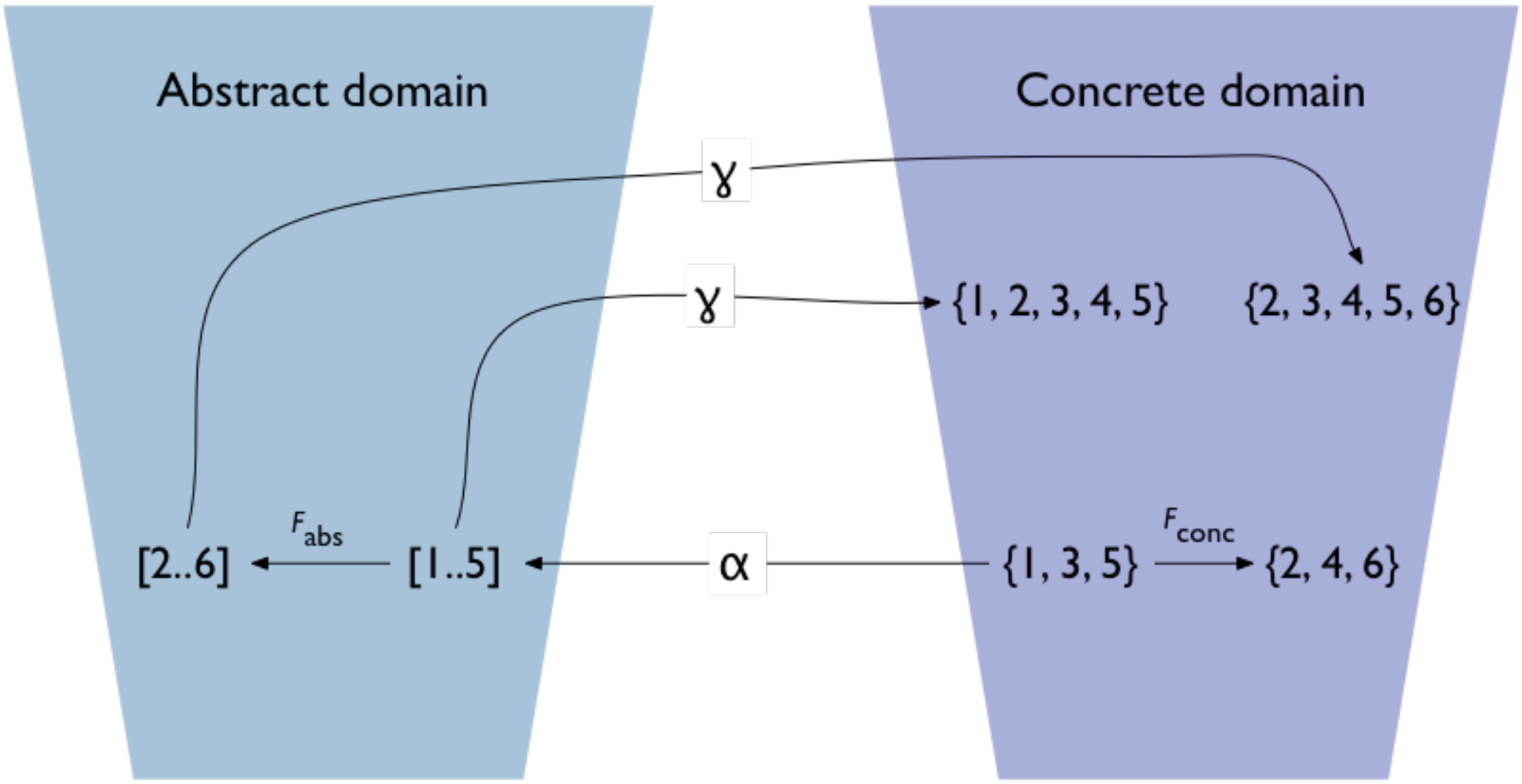
Goal for the summer

- **Find out how complete and precise our IA32 TSL specification is...**
  - ...by generating an IA32 emulator, then comparing the emulator to the real processor.
    - If resulting states differ on the same inputs, the spec was (*probably)* buggy.
- *We already have all the pieces*: IA32 spec, EMUL, and a third-party tool for testing CPU emulators. This will be easy, right?!

A brief digression: abstract interpretation, γ, and α



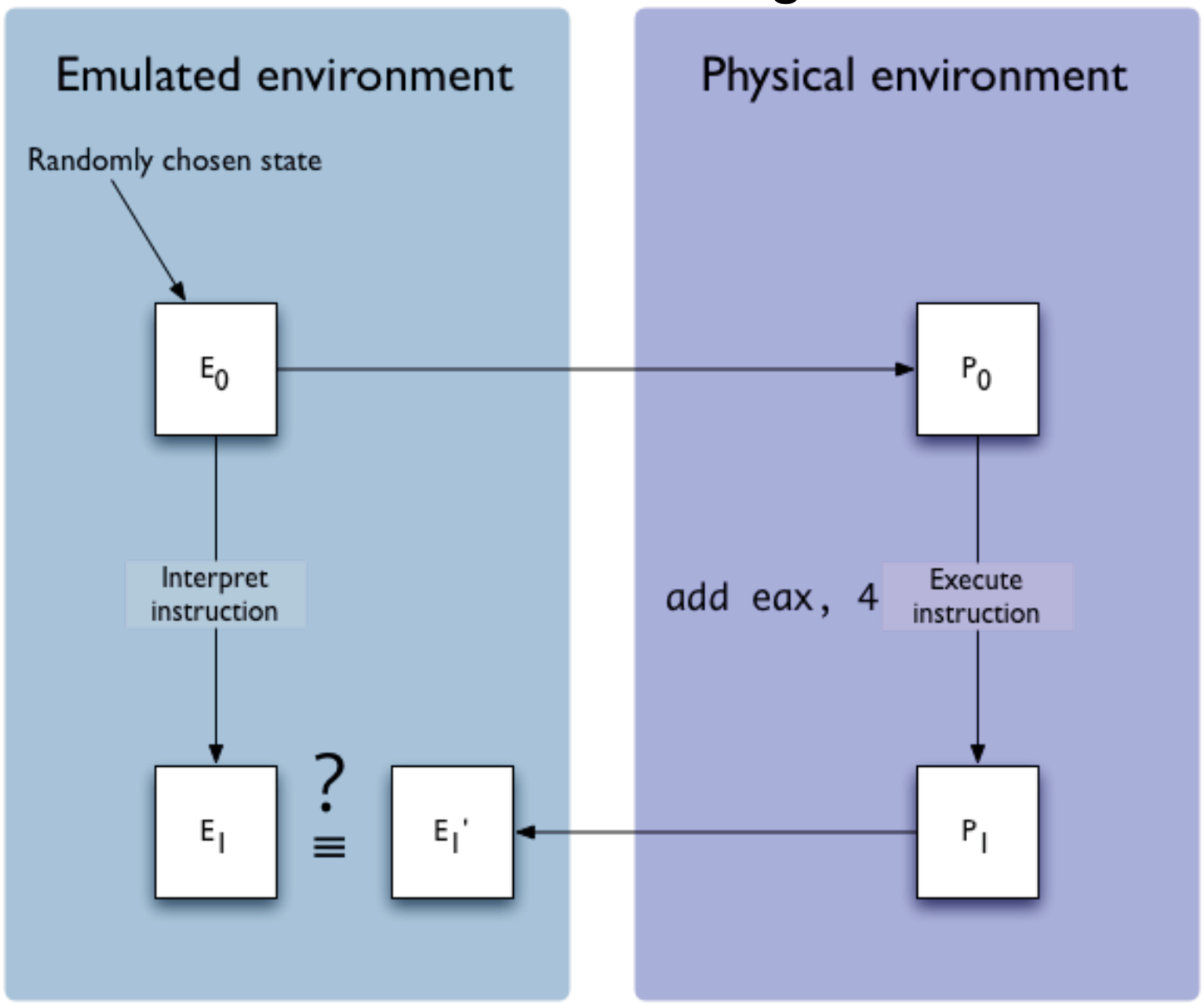
A brief digression: abstract interpretation, γ, and α



How to test a CPU emulator

How to test a CPU emulator

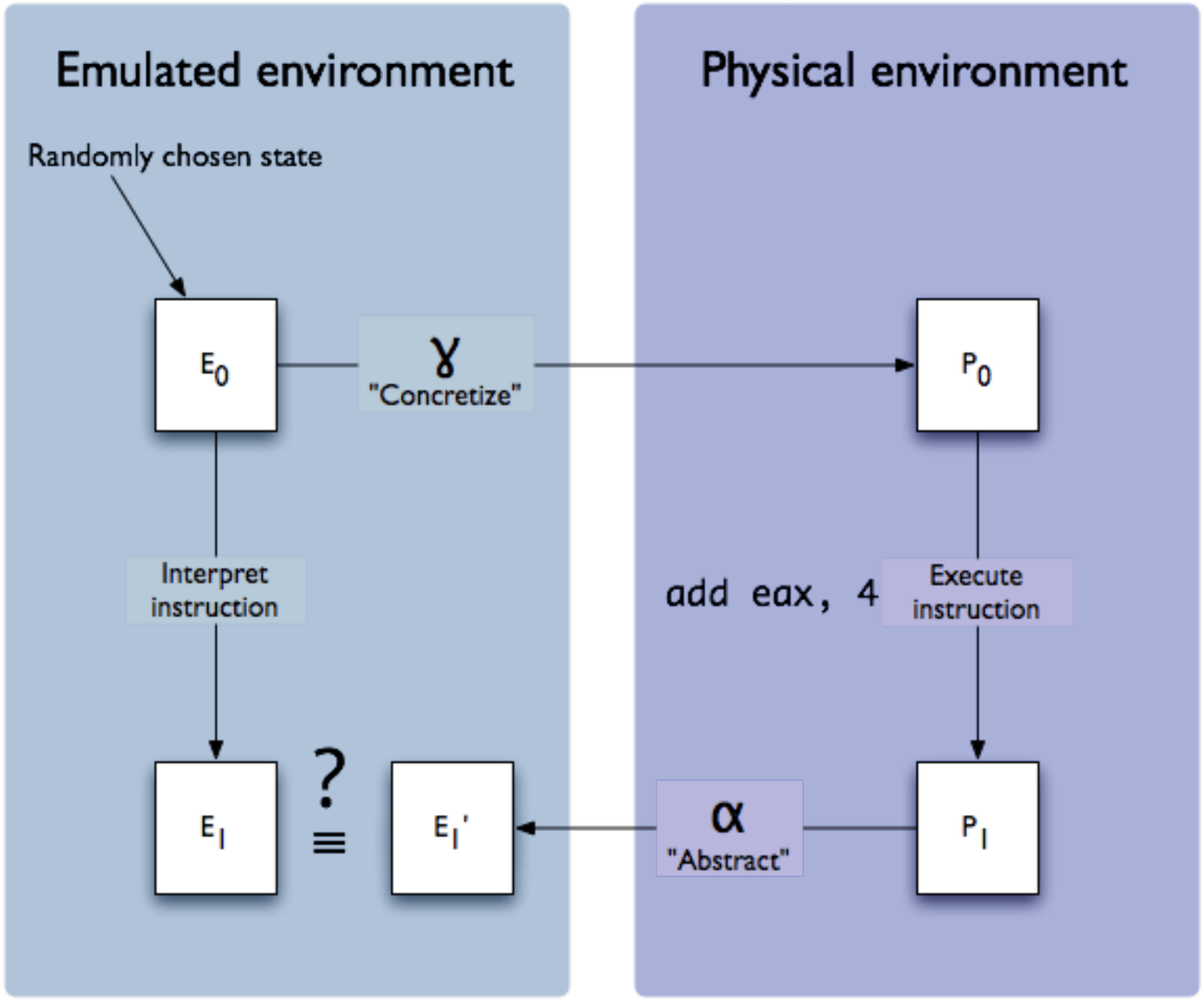
EmuFuzzer's design



Martignioni, L. *et al.*, "Testing CPU Emulators", ISSTA '09

How to test a CPU emulator

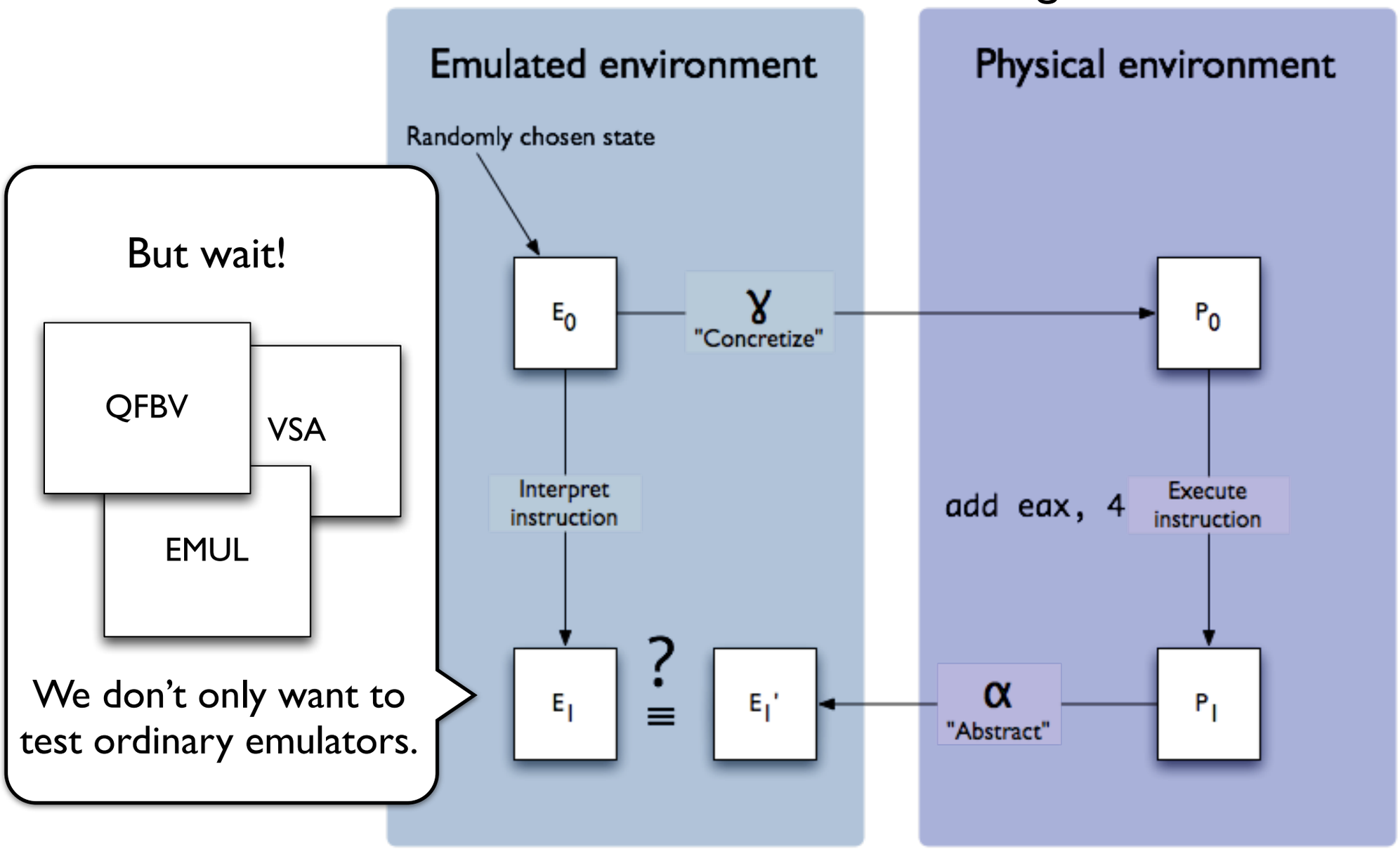
EmuFuzzer's design



Martignioni, L. *et al.*, "Testing CPU Emulators", ISSTA '09

How to test a CPU emulator

EmuFuzzer's design



Martignioni, L. *et al.*, "Testing CPU Emulators", ISSTA '09

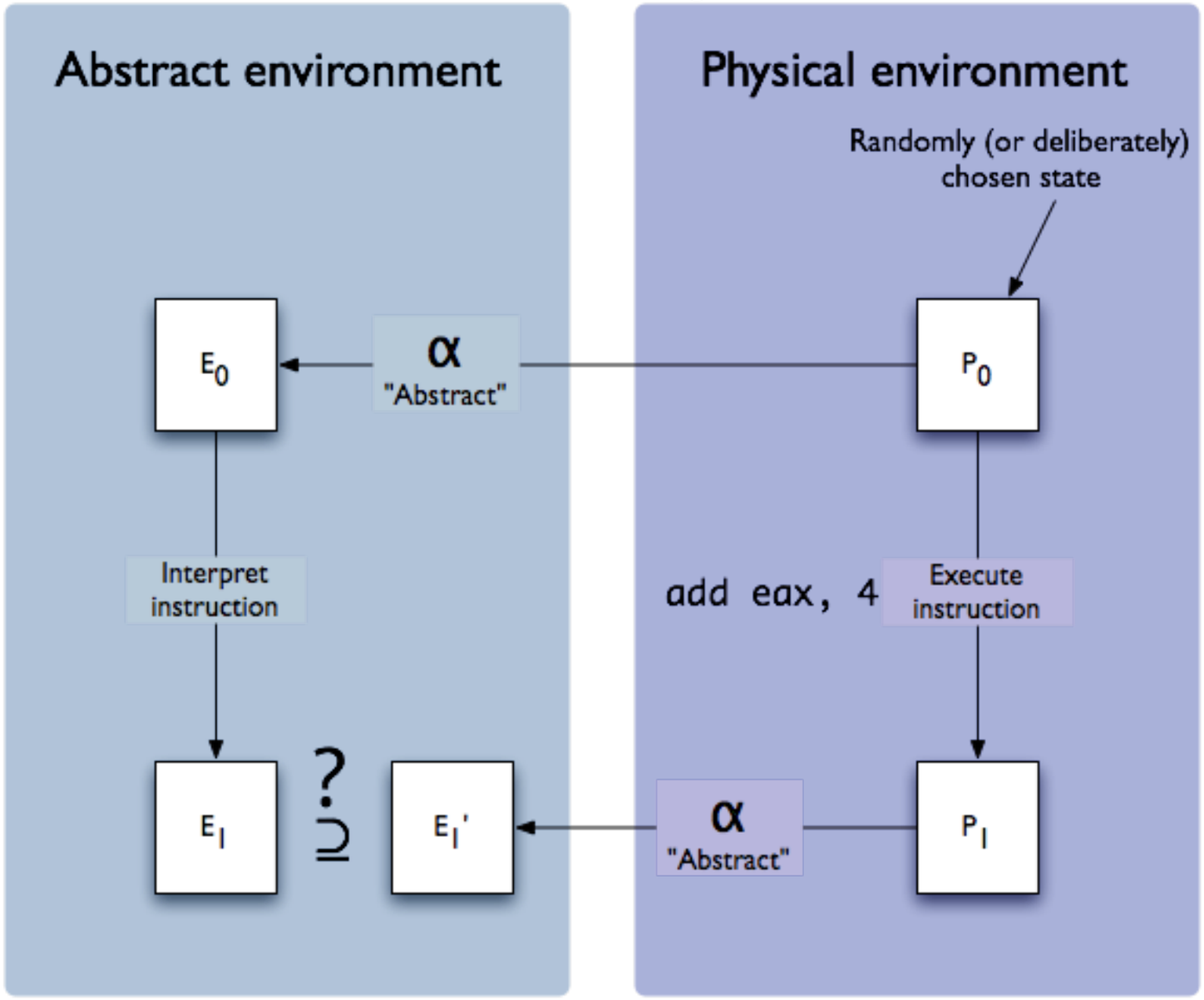
(someday)

How to test any analysis engine

(someday)

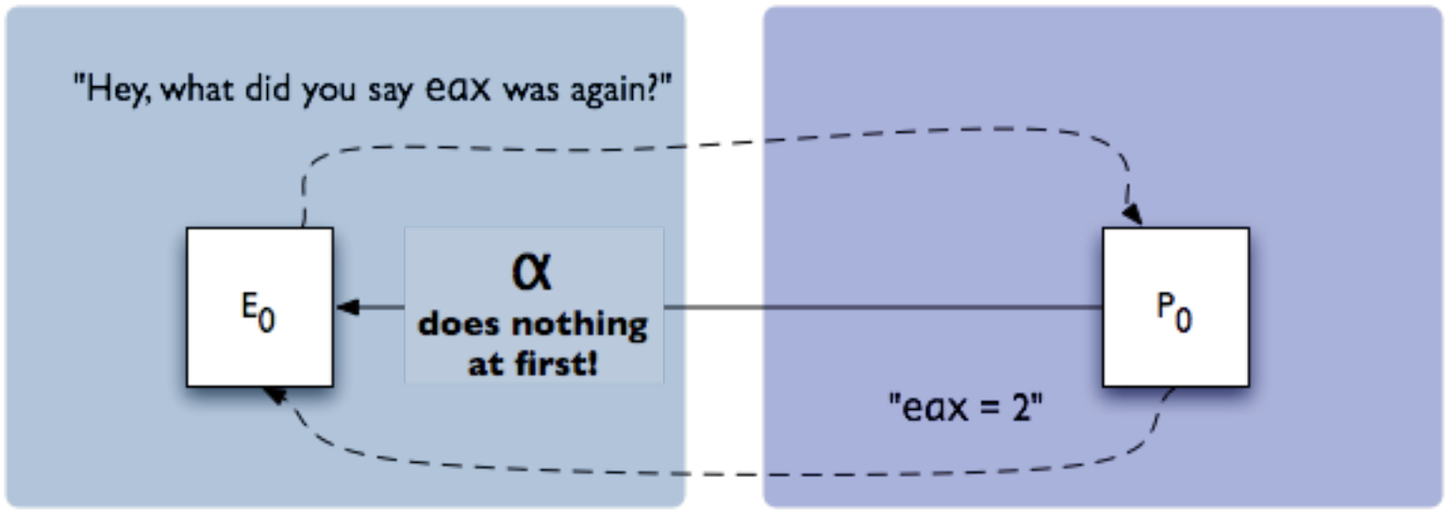
How to test any analysis engine

Our design

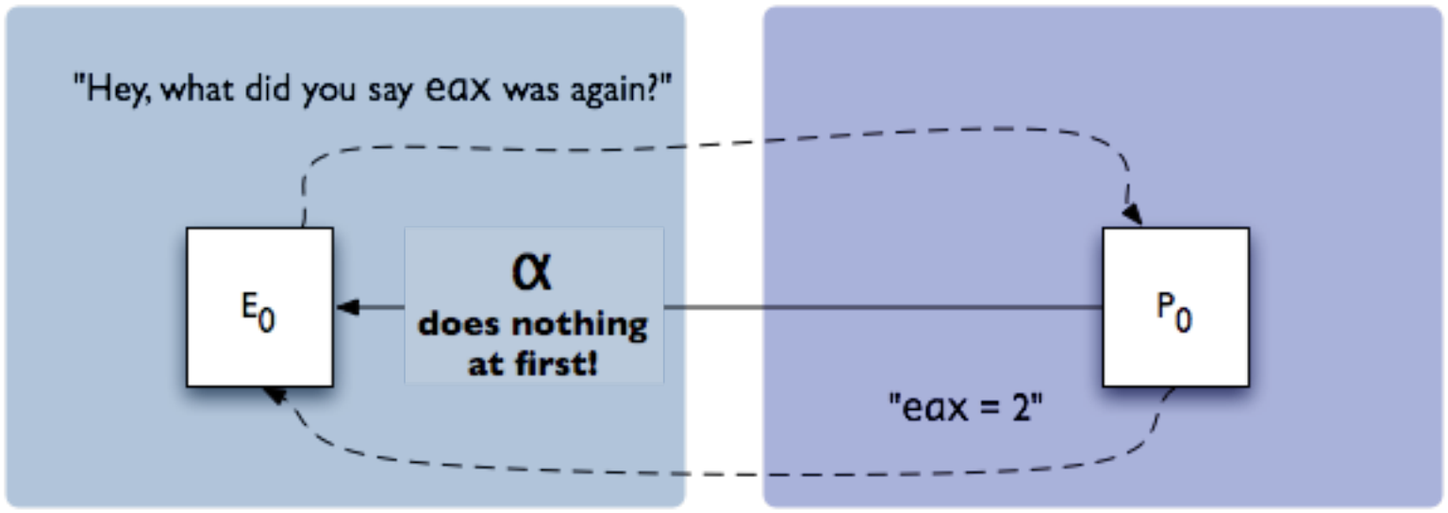


Look-thru memory

Look-thru memory



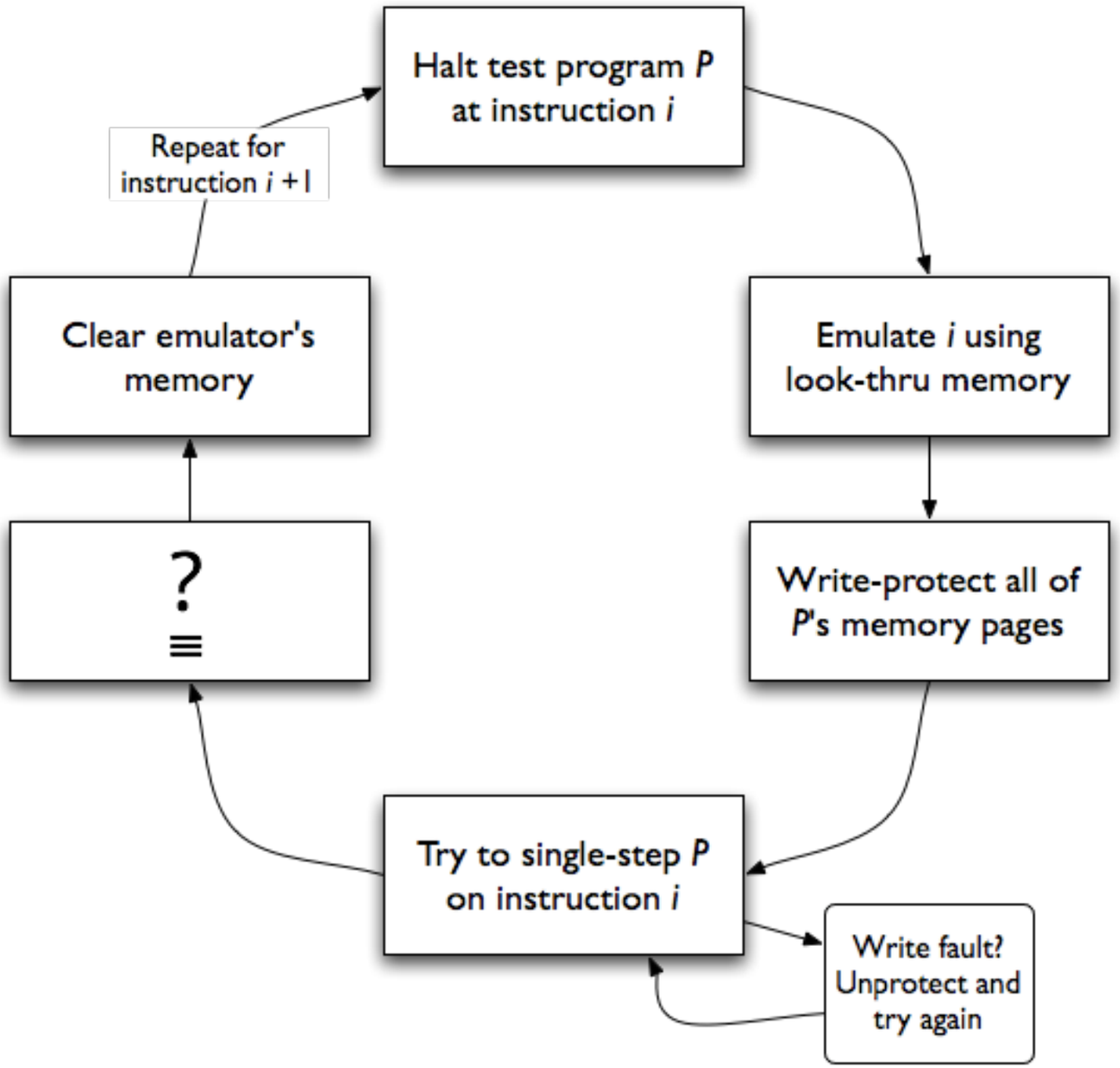
Look-thru memory



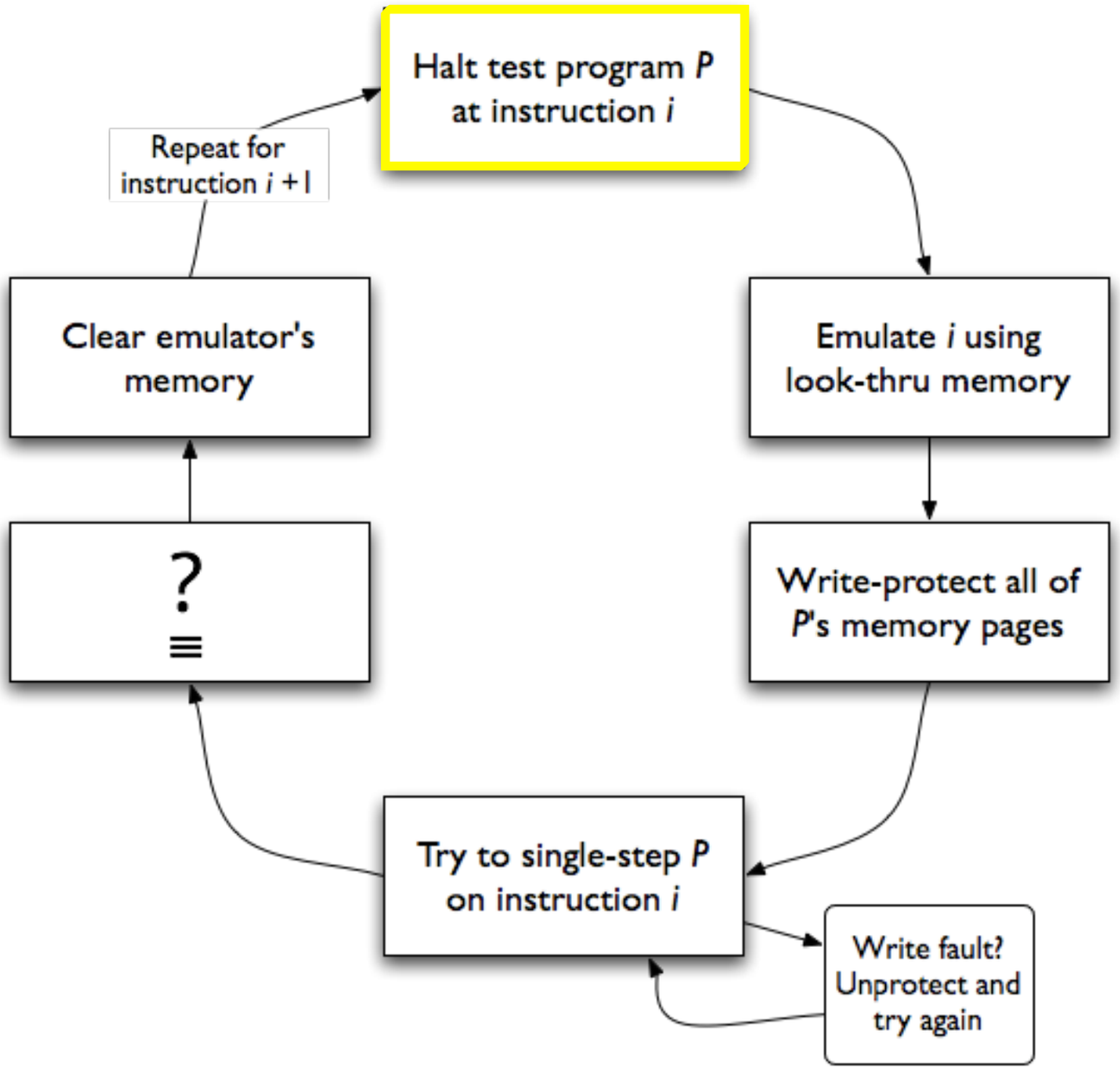
- The ability to *lazily* instantiate the emulator's state (memory and registers) from that of the process as each instruction is being emulated.

TSL validator main loop

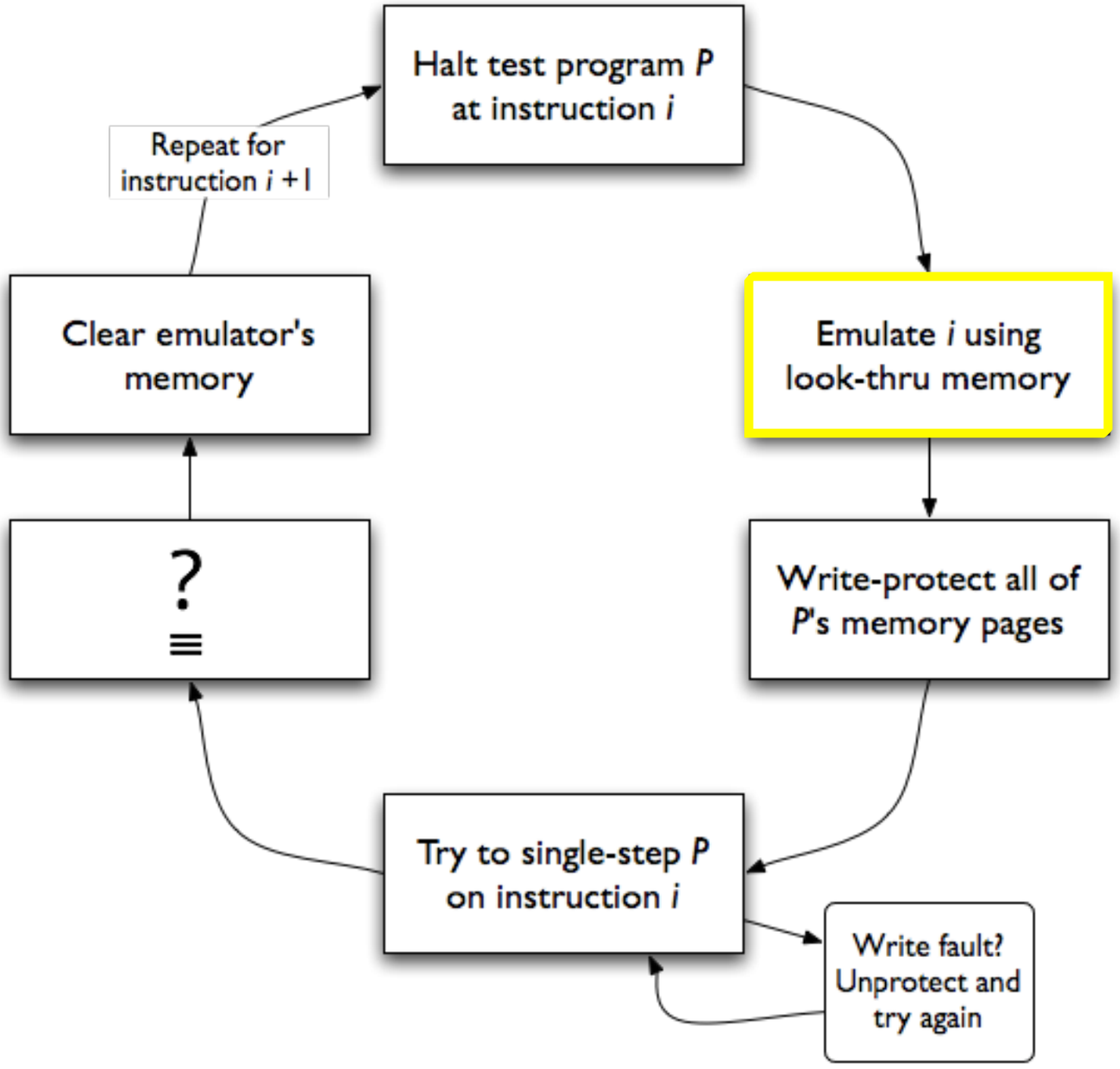
TSL validator main loop

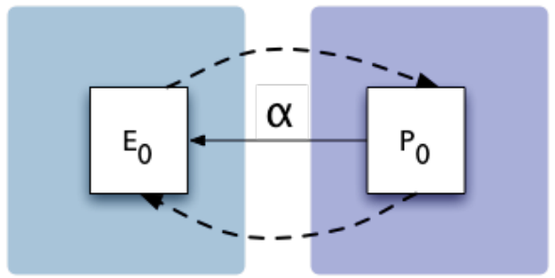


TSL validator main loop

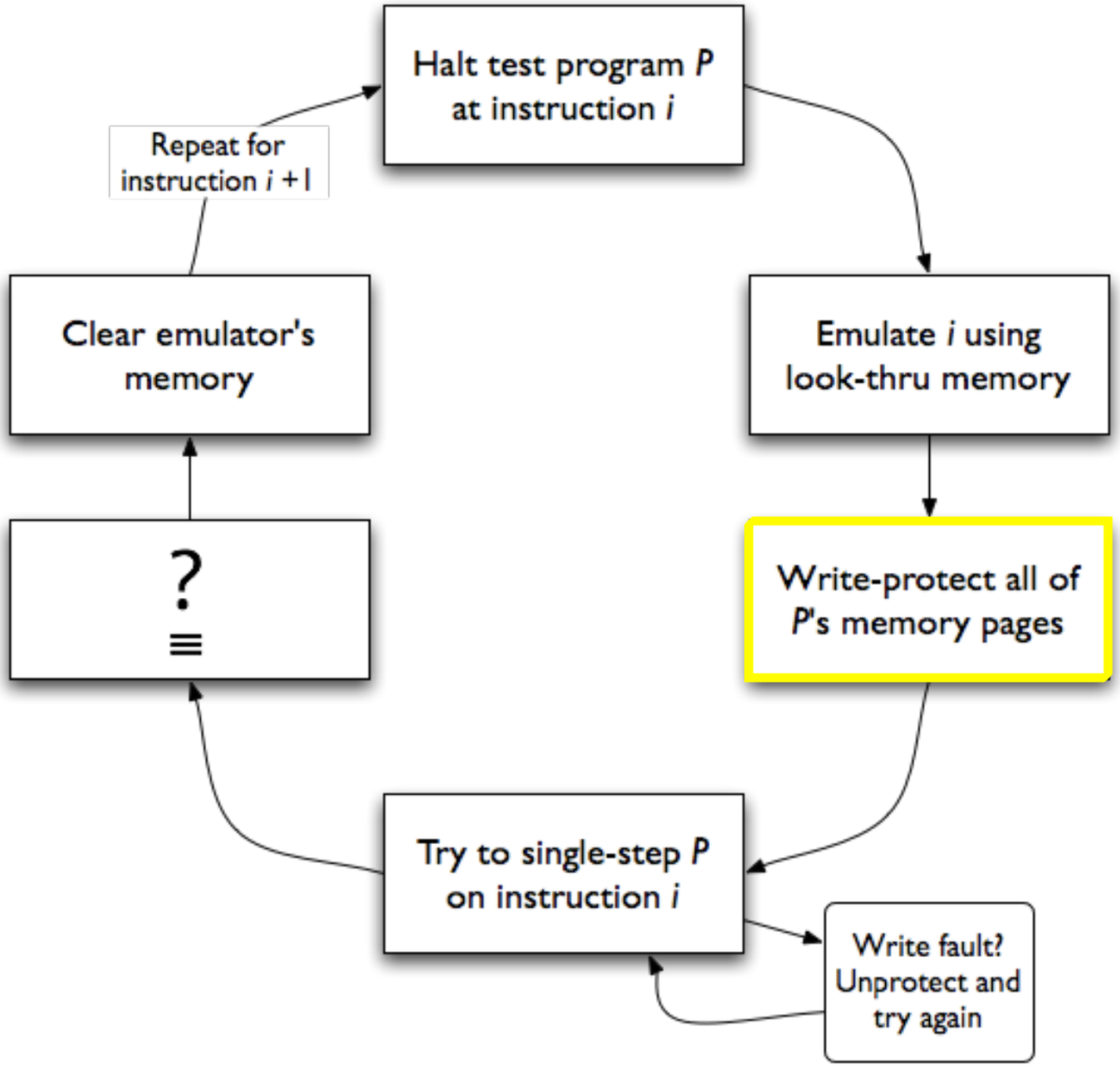


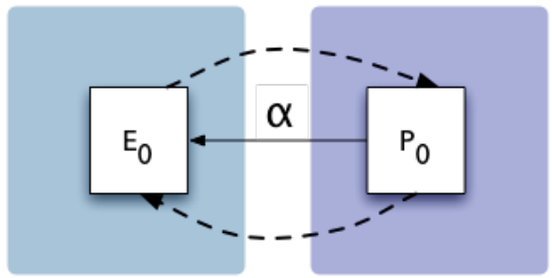
TSL validator main loop



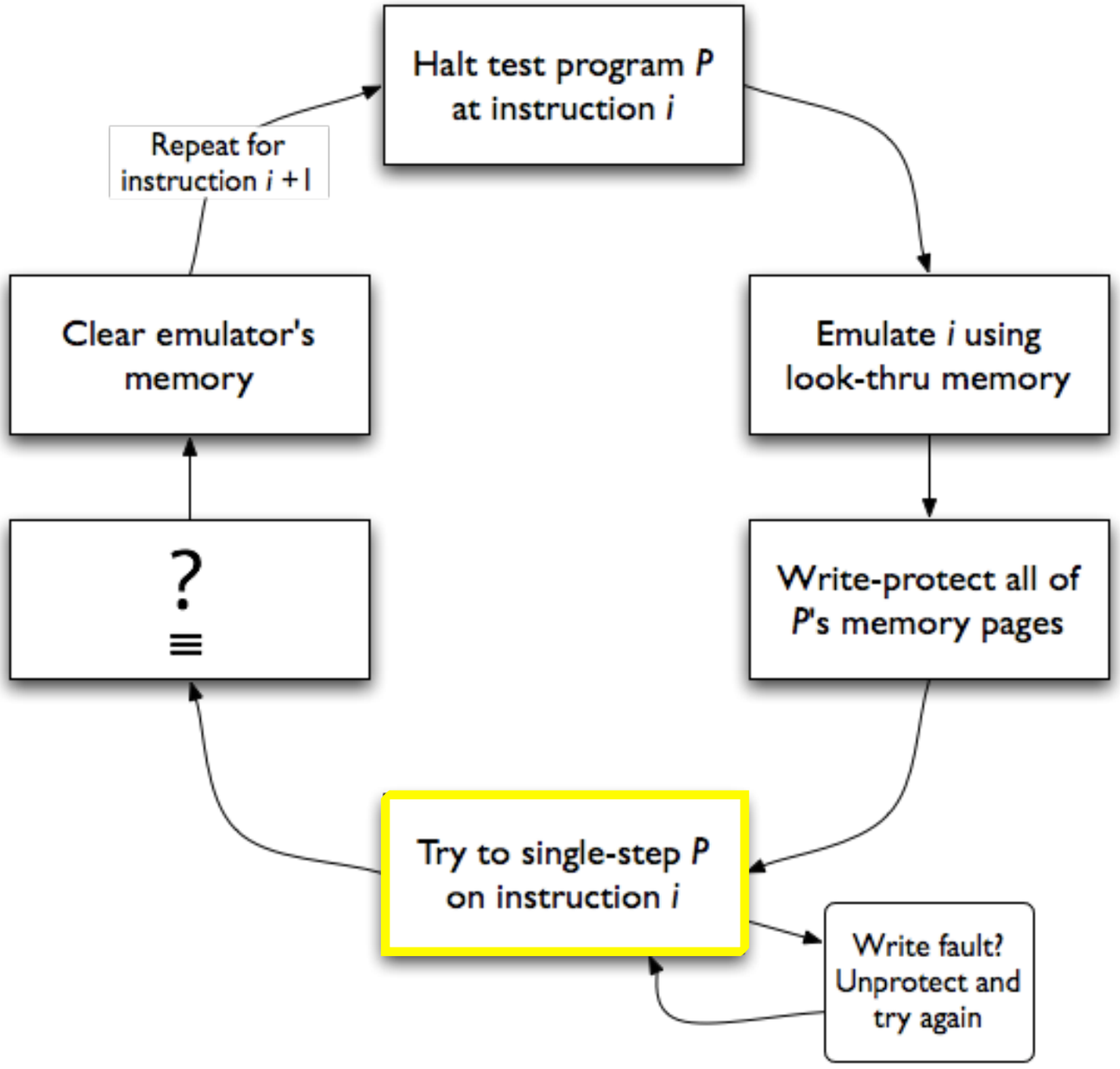


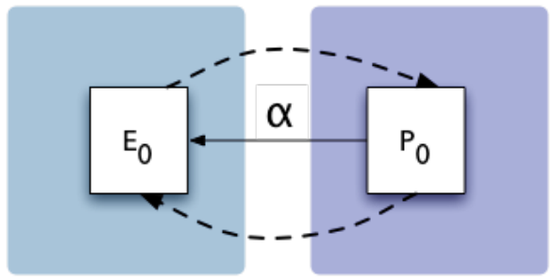
TSL validator main loop



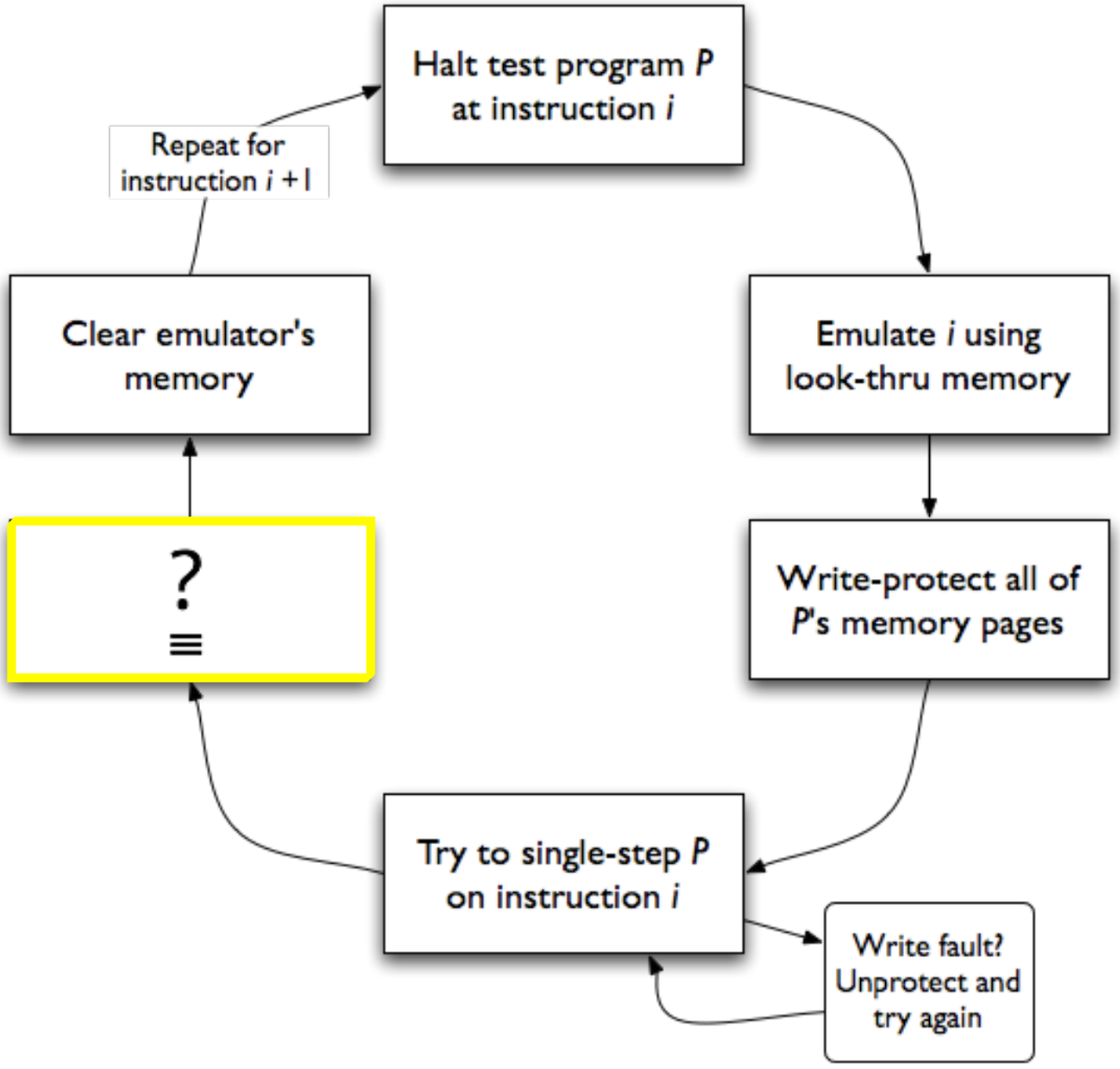


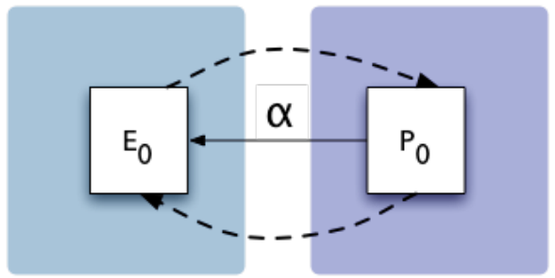
TSL validator main loop



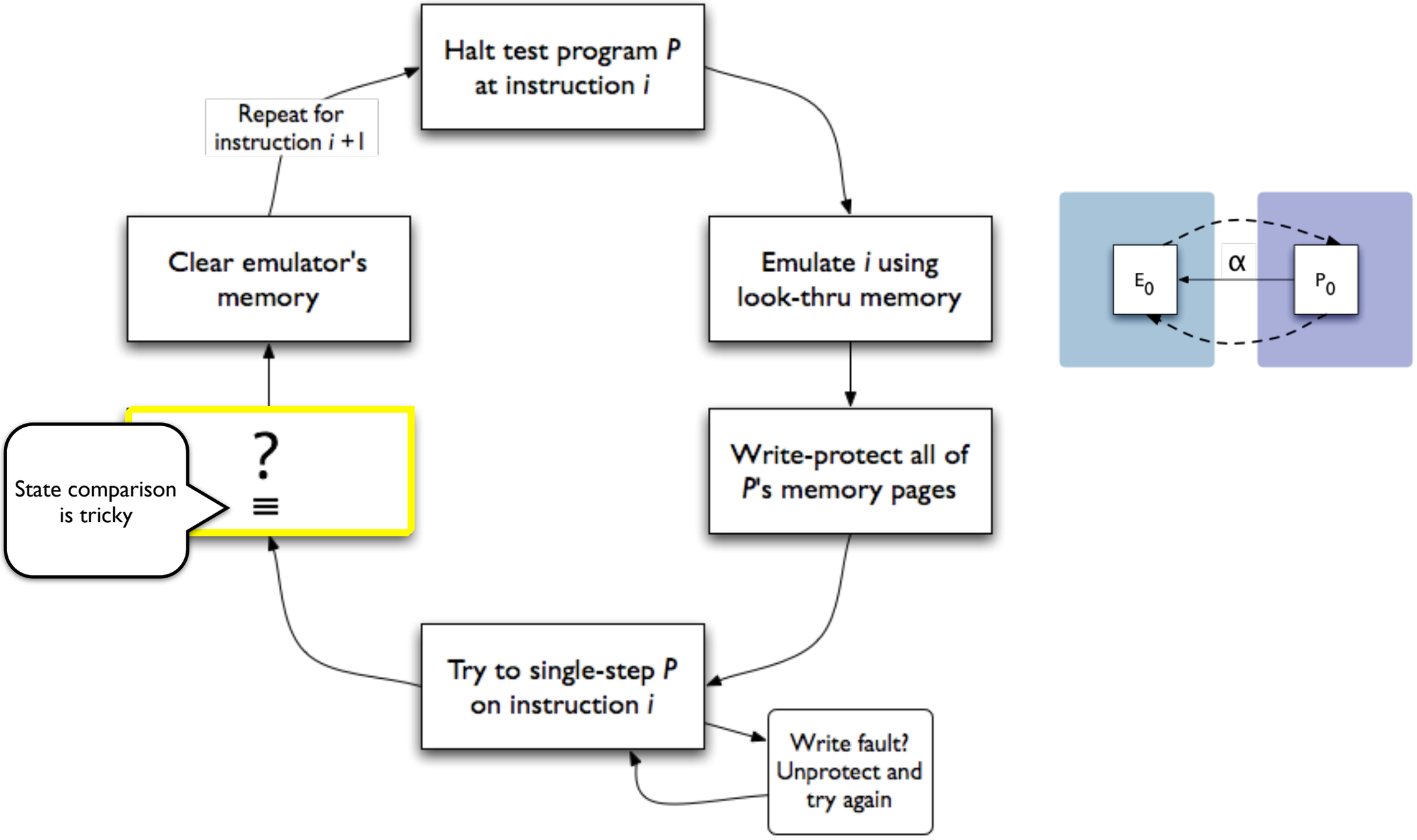


TSL validator main loop

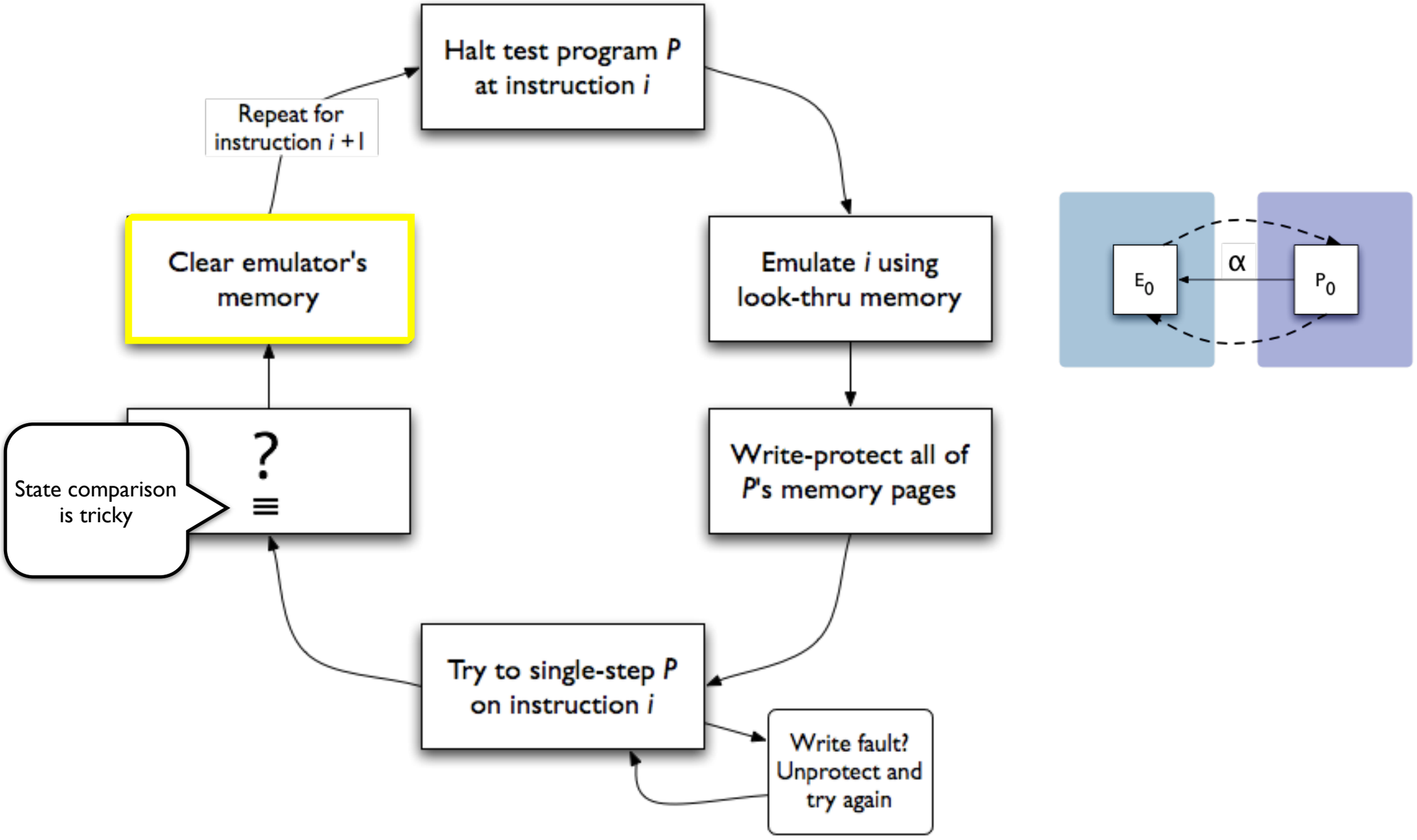




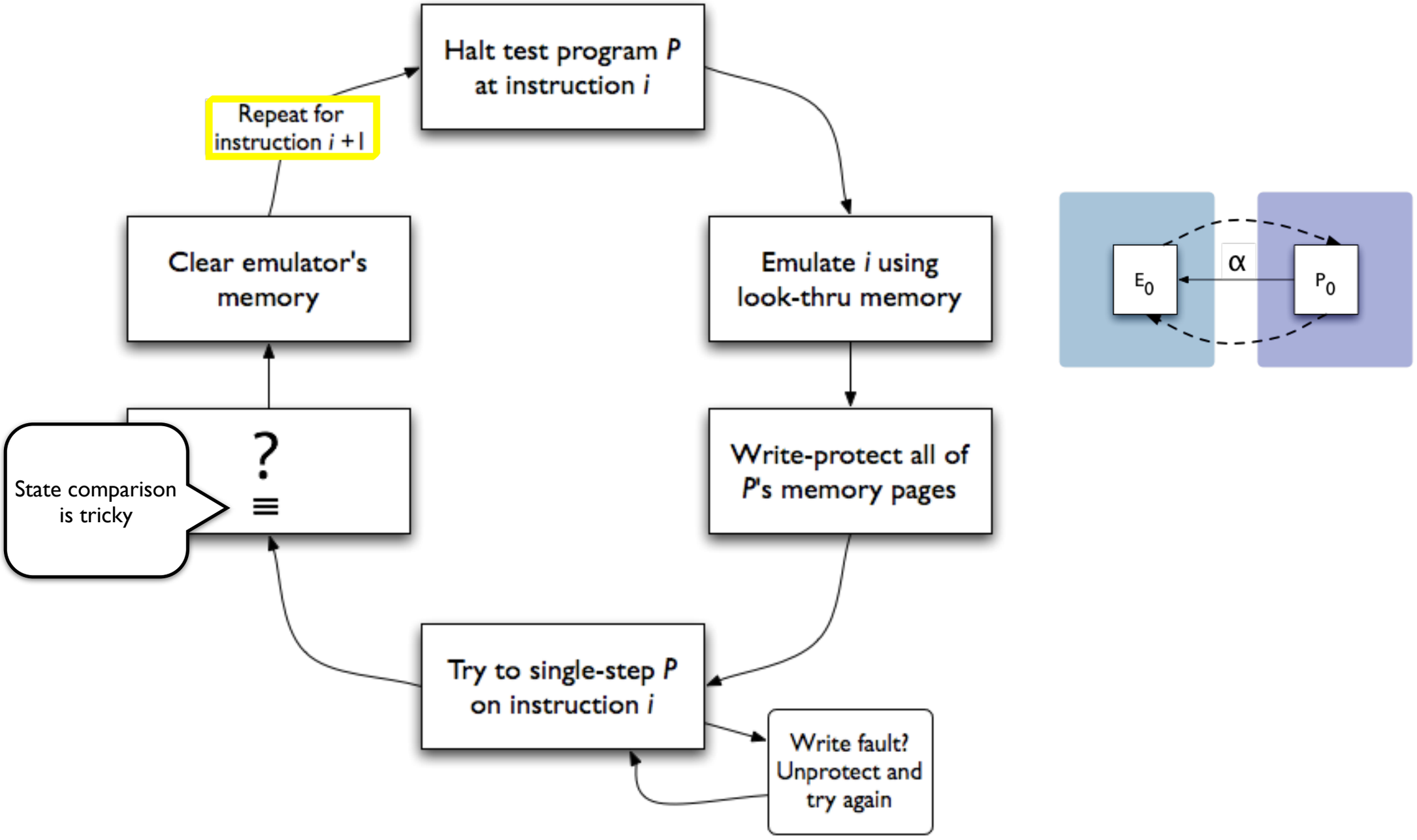
TSL validator main loop



TSL validator main loop



TSL validator main loop



Future work: short-term stuff

Future work: short-term stuff

- **The hard part of state comparison**: identify changed locations on the real process side, and compare them with corresponding locations on the emulator side.

Future work: short-term stuff

- **The hard part of state comparison**: identify changed locations on the real process side, and compare them with corresponding locations on the emulator side.

- Better logging and reporting: eventually, we'd like to have a "dashboard" that will tell us roughly how complete and correct the existing TSL specifications are.

Future work: short-term stuff

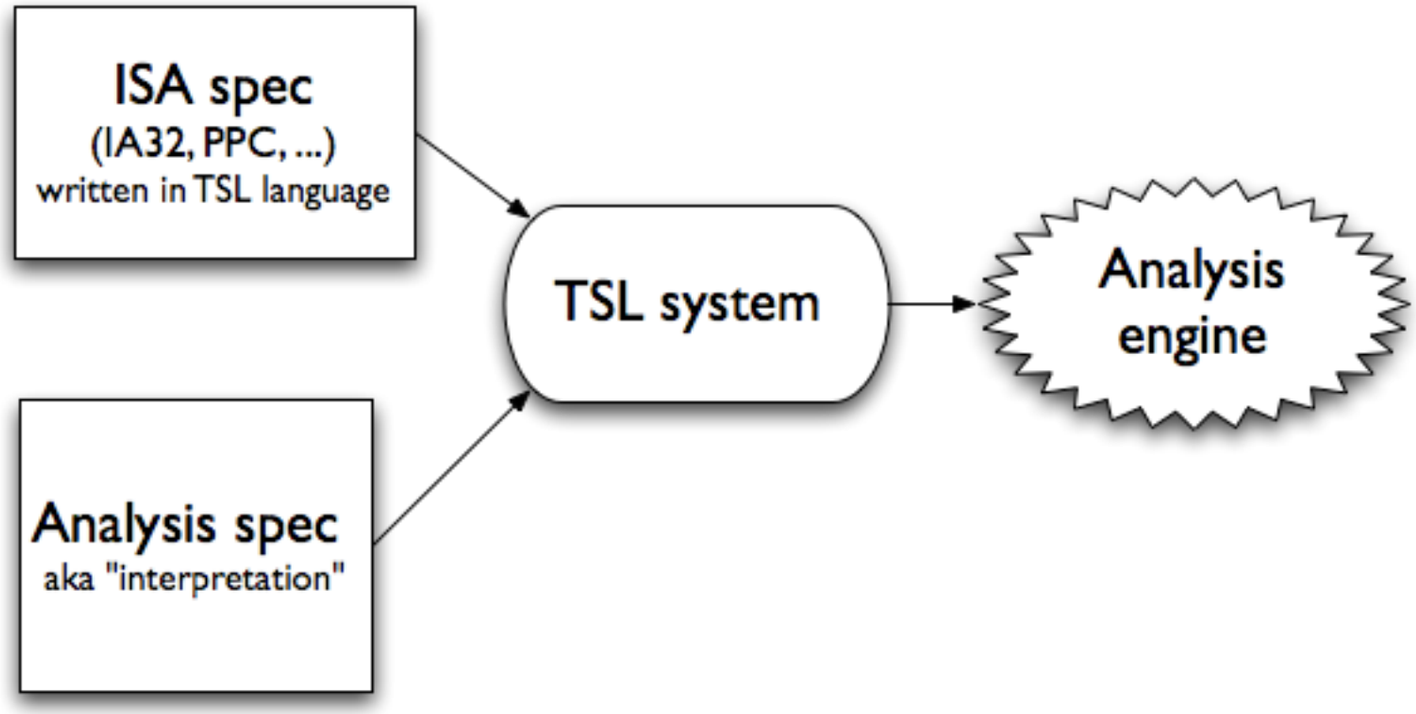
- **The hard part of state comparison**: identify changed locations on the real process side, and compare them with corresponding locations on the emulator side.

- Better logging and reporting: eventually, we'd like to have a "dashboard" that will tell us roughly how complete and correct the existing TSL specifications are.

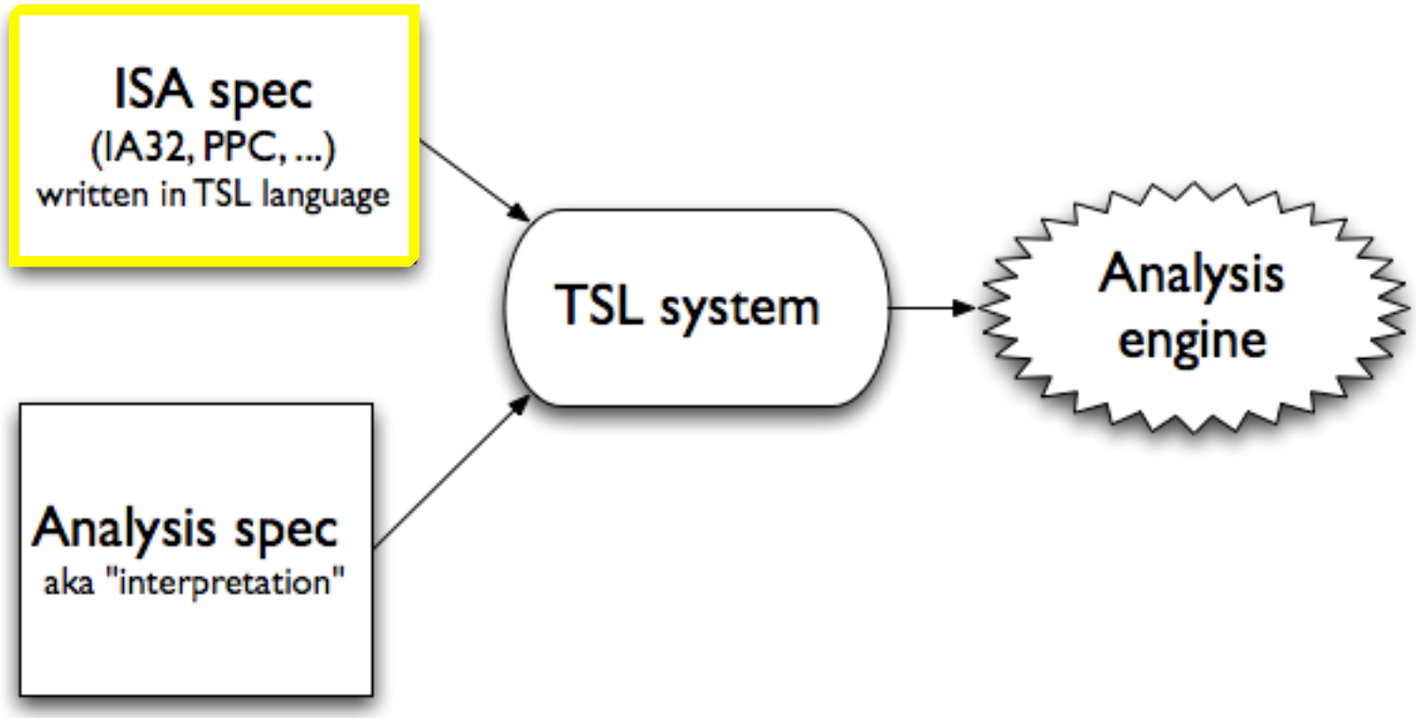
- How to deal with test programs that "misbehave"? (Programs that mess with their own memory protection, install their own seg fault handlers, ...)

Future work: long-term stuff

Future work: long-term stuff

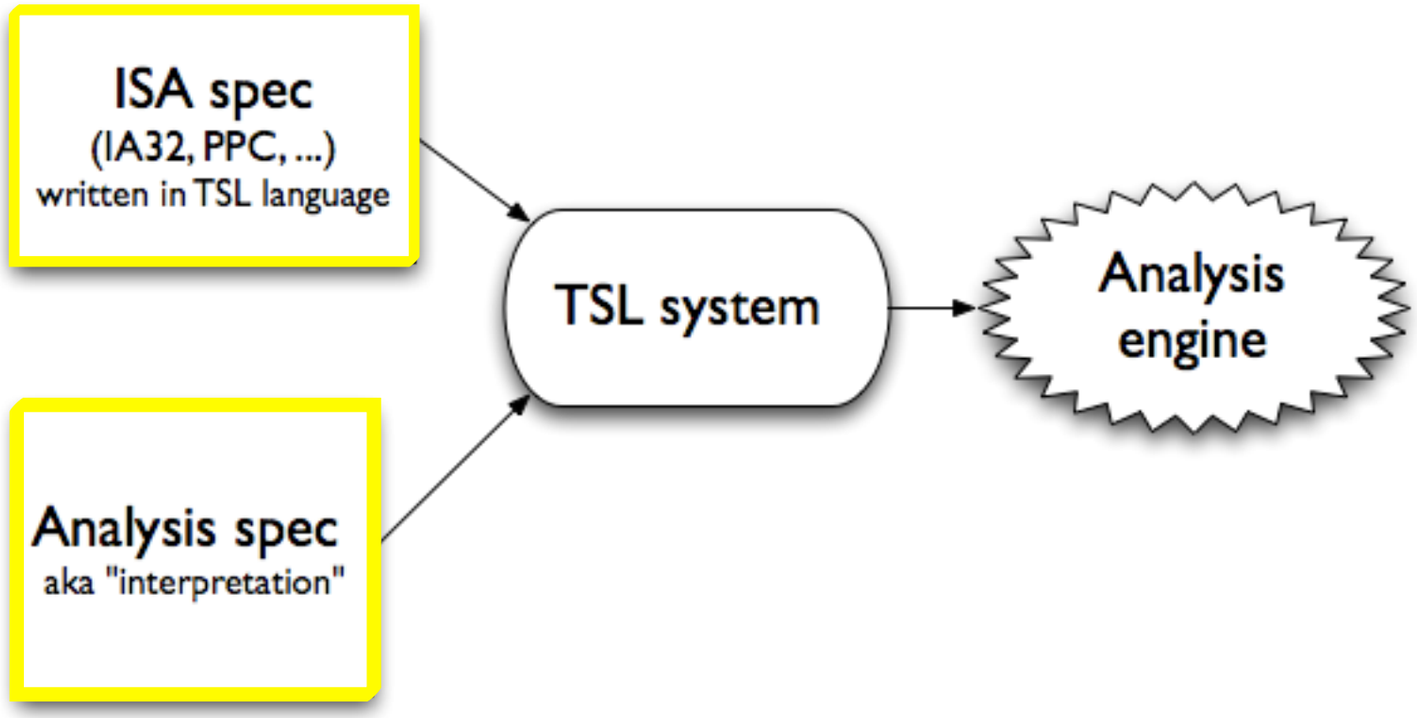


Future work: long-term stuff



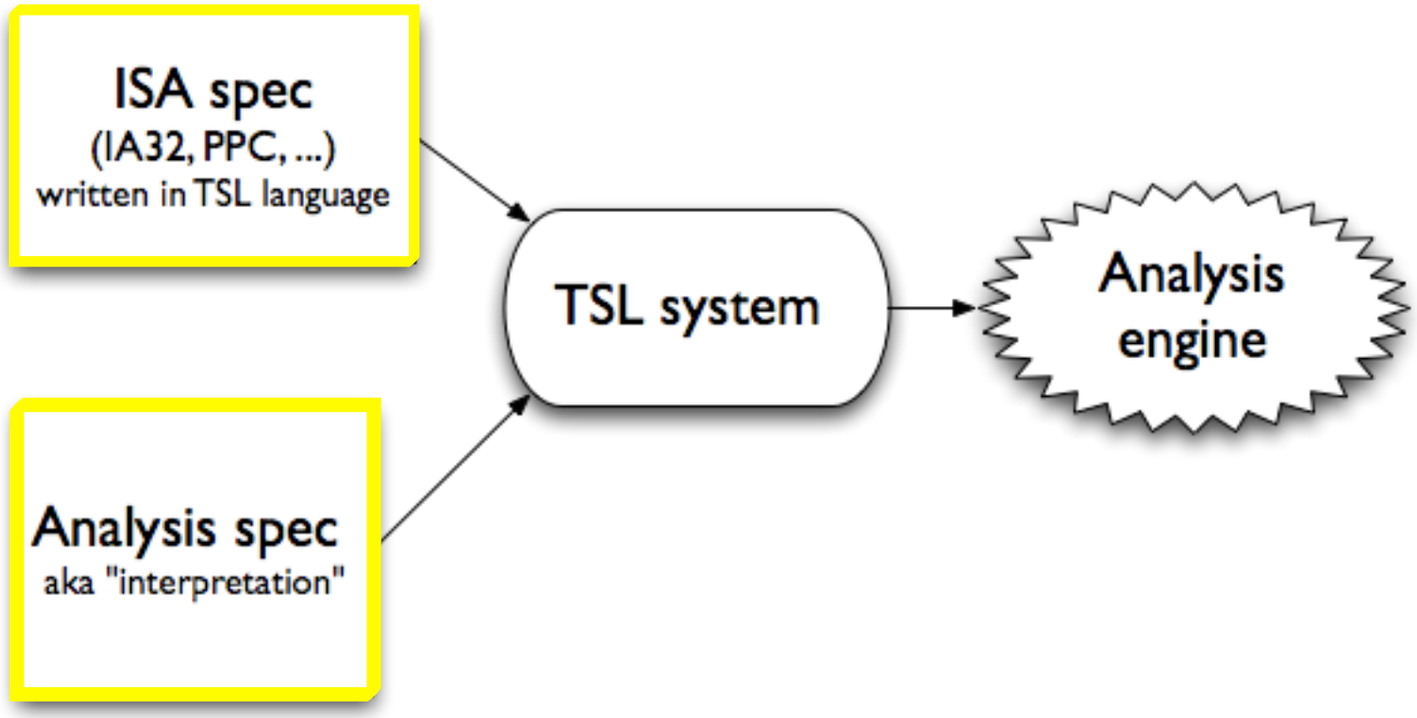
- Support for more ISAs. (x64, at least!)

Future work: long-term stuff



- Support for more ISAs. (x64, at least!)
- **Support for true abstract interpretations**, not just EMUL.

Future work: long-term stuff



- Support for more ISAs. (x64, at least!)
- **Support for true abstract interpretations**, not just EMUL.
- Find ways to choose which inputs to test that will be most likely to turn up bugs in a specification.

What I learned

What I learned

- Emulators, debuggers, and static analyzers are not made of magic

What I learned

- Emulators, debuggers, and static analyzers are not made of magic
- First real systems programming experience: didn't quite cross the kernel space boundary, but came right up next to it

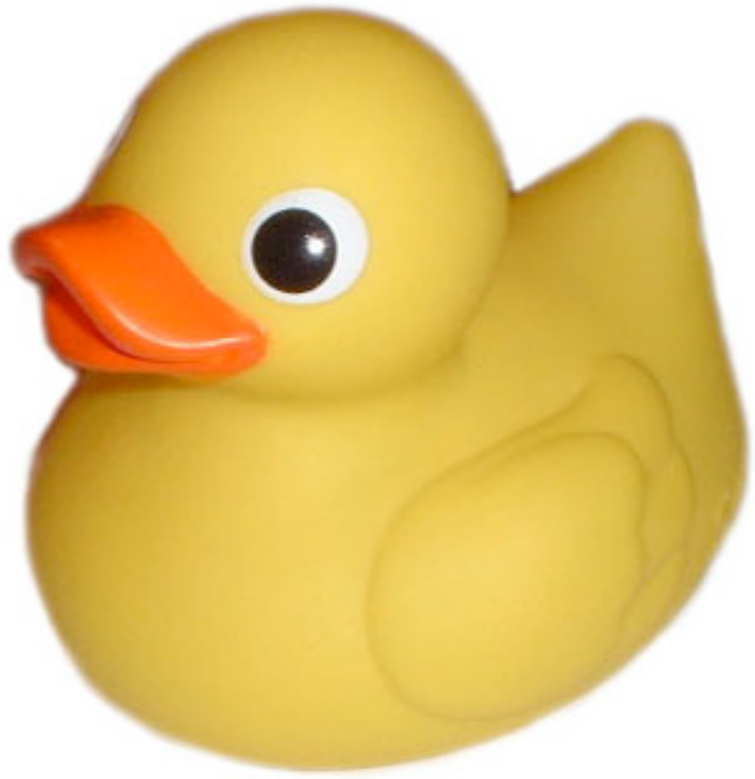
What I learned

- Emulators, debuggers, and static analyzers are not made of magic
- First real systems programming experience: didn't quite cross the kernel space boundary, but came right up next to it
- A metric for how much I can accomplish in 13 weeks

What I learned

- Emulators, debuggers, and static analyzers are not made of magic
- First real systems programming experience: didn't quite cross the kernel space boundary, but came right up next to it
- A metric for how much I can accomplish in 13 weeks
- Finally convinced that OOP is good for something

Thank you!



Questions?

(exit)