TransForm: Formally Specifying Transistency Models and Synthesizing Enhanced Litmus Tests

Naorin Hossain
Princeton University

Caroline Trippel
Stanford University

Margaret Martonosi
Princeton University

ISCA 2020

Memory Consistency Models (MCMs) are used to specify legal memory access orderings

- MCMs specify rules for legal values that can be returned when software loads from memory on a shared memory system

Problem: can't reason about program executions impacted by shared virtual memory (VM) state with existing MCM constructs

Ini

W_0 x = 1	R_2 y = **?**
W_1 y = 1	R_3 x = **?**

Litmus test: small diagnostic

W_1 y = 1	R_3 x = **1**

Core 0	Core 1

Core 0	Core 1

This work: Memory Transistency Models (MTMs) – the superset of MCMs that additionally capture VM-aware ordering specifications

How does VM affect consistency?

Virtual-to-physical address (VA-to-PA) mappings are stored in **page table entries (PTEs)**…

...and cached in the **translation lookaside buffer (TLB).**

Page table

Status bits				VA-to-PA mapping
A	D	R	W	VA x → PA a
A	D	R	W	VA y → PA b

.
.
.

Read y →

TLB

VA-to-PA mapping
VA x → PA a
VA y → PA b

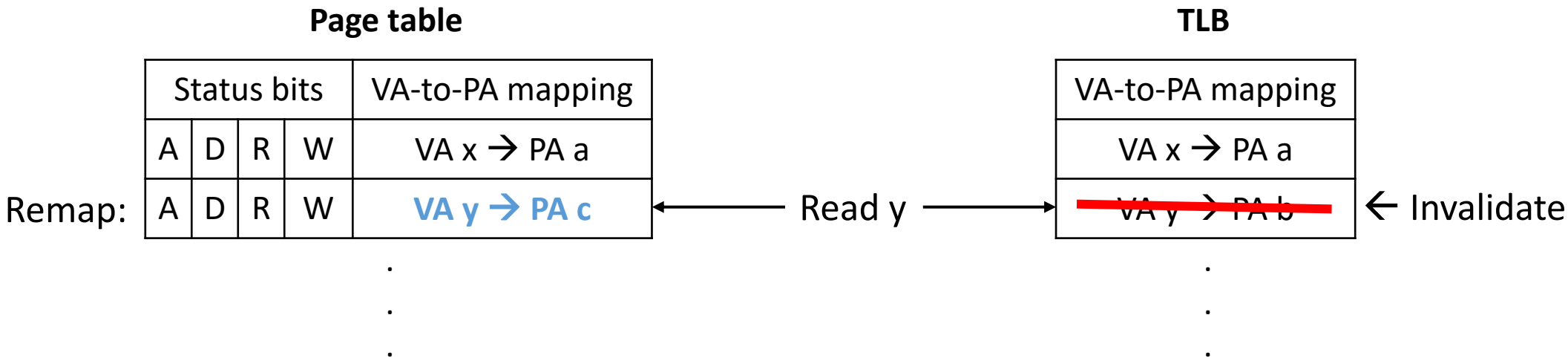
.
.
.

When a mapping is changed in the page table, corresponding TLB entries must be **invalidated**.

How does VM affect consistency?

Virtual-to-physical address (VA-to-PA) mappings are stored in **page table entries (PTEs)**…

...and cached in the **translation lookaside buffer (TLB).**

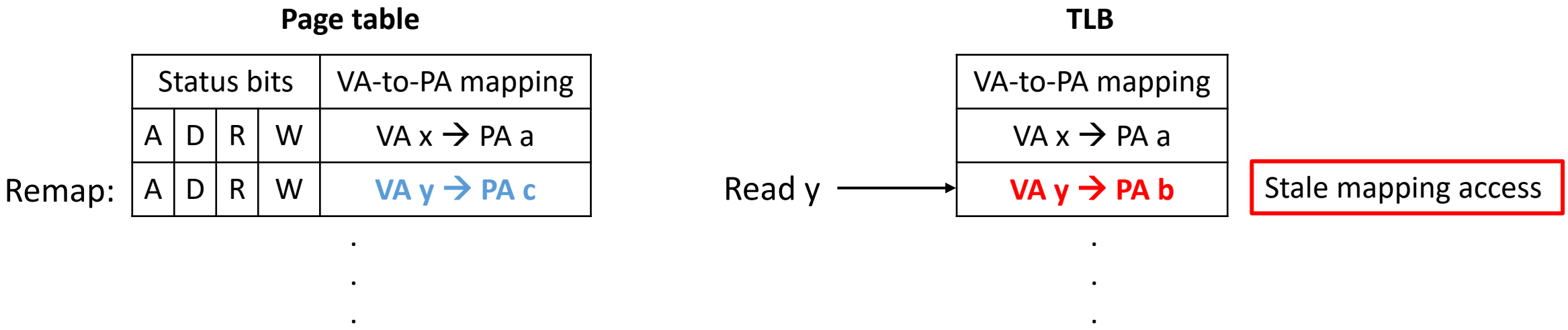


When a mapping is changed in the page table, corresponding TLB entries must be **invalidated**.

How does VM affect consistency?

Virtual-to-physical address (VA-to-PA) mappings are stored in **page table entries (PTEs)**…

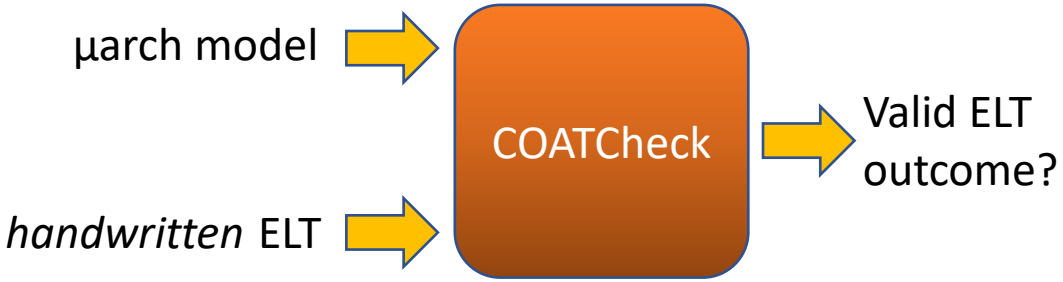
...and cached in the **translation lookaside buffer (TLB).**

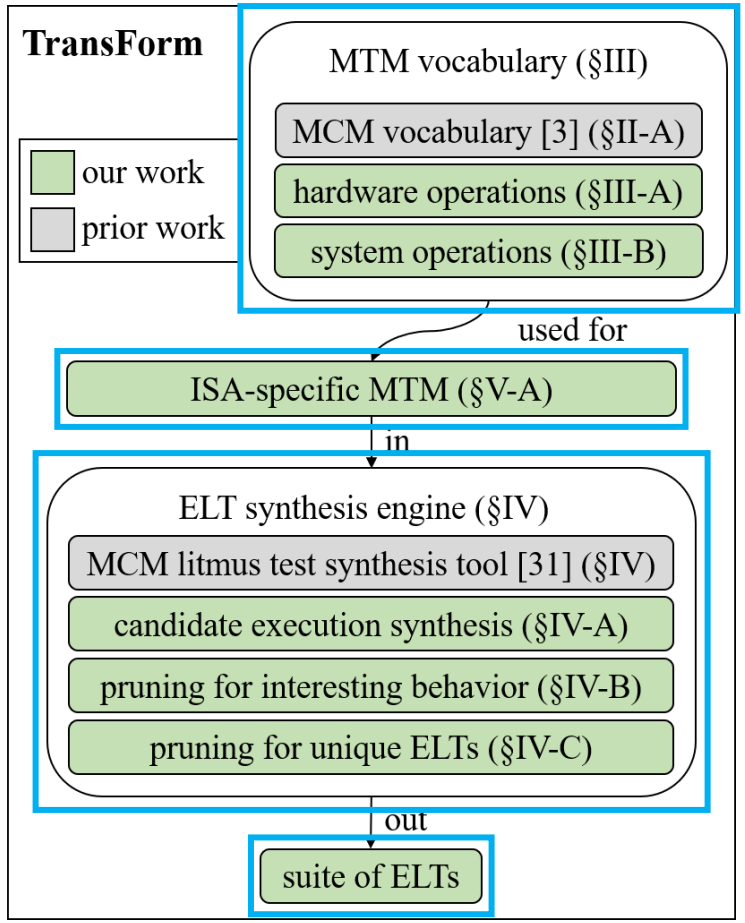


AMD Athlon™ 64 and AMD Opteron™ Processor bug: INVLPG (x86 TLB entry invalidation instruction) fails to invalidate TLB entry in certain cases

TransForm introduces constructs for ISA-level MTM specification and ELT synthesis

- Formal MTM vocabulary captures system- and hardware-level VM events and interactions with user-facing program instructions
- Enables ISA-level MTM specification
- Enables automated *enhanced litmus test (ELT)* synthesis





Automated!

Outline

- Background on ISA-level MCM vocabulary
- Introduction to ISA-level MTM vocabulary
- Automating synthesis of ELTs
- Case Study: an estimated MTM for x86
- Conclusions

ISA-level MCM relations can describe programs and their *candidate executions*

Program

Core 0
W_0 x
R_1 x
W_2 x

Candidate execution

Core 0
W_0 x = 1
R_1 x = 1
W_2 x = 2

Instructions

Event = {W_0, R_1, W_2}

MemoryEvent = {W_0, R_1, W_2}

Location = {x}

address {W_0 → x, R_1 → x, W_2 → x}

program order (po)

po = {W_0 → R_1, R_1 → W_2}

Communication (com) relations

reads from (rf)

rf = {W_0 → R_1}

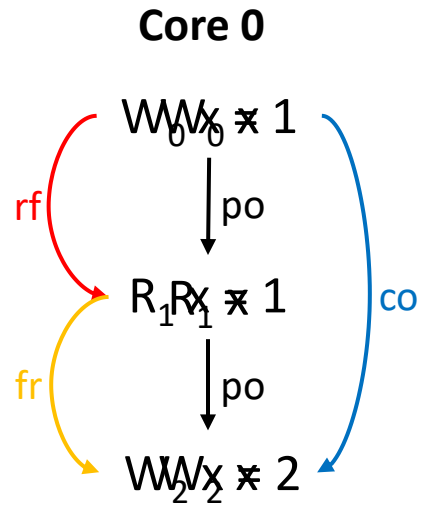
coherence order (co)

co = {W_0 → W_2}

from reads (fr)

fr = {R_1 → W_2}

Graph



Accessed data (outcome) symbolically represented by com relations

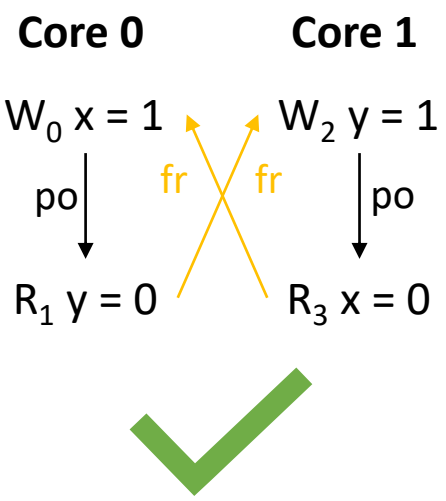
MCM specifications place constraints on permitted execution behaviors

Consistency predicates constrain candidate execution behavior based on MCM specifications

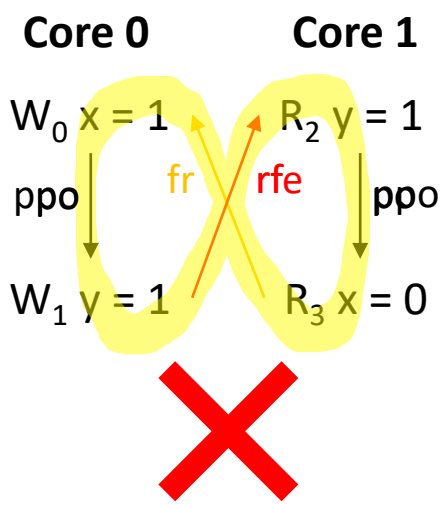
Intel x86 processors use the **total store order (TSO)** memory model (**x86-TSO**)

Causality – axiom in x86-TSO consistency predicate:
acyclic(rfe + co + fr + ppo + fence)

sb ("store buffering") litmus test



mp ("message passing") litmus test



Augmenting MCMs to include MTM features

What MCMs have	What is needed for MTM interactions
Support for static VA-to-PA mappings without aliasing.	Support for VA-to-PA mappings that can be modified during a program's execution.
Support for user-level instruction interactions through shared memory.	Support for shared memory interactions between user-level instructions and system- and hardware-level operations.

transistency operations

Outline

- Background on ISA-level MCM vocabulary
- Introduction to ISA-level MTM vocabulary
- Automating synthesis of ELTs
- Case Study: a estimated MTM for x86
- Conclusions

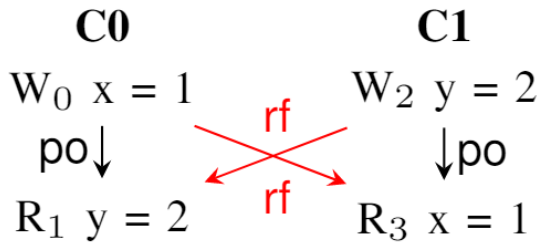
MTM Vocabulary: Hardware-level operations

TransForm supports **page table walks (PT walks)** and **dirty bit updates**

Ghost instructions

PT walk: loads translation lookaside buffer (TLB) entry

dirty bit update: modifies dirty bit in page table entry (PTE)



MTM Vocabulary: Hardware-level operations

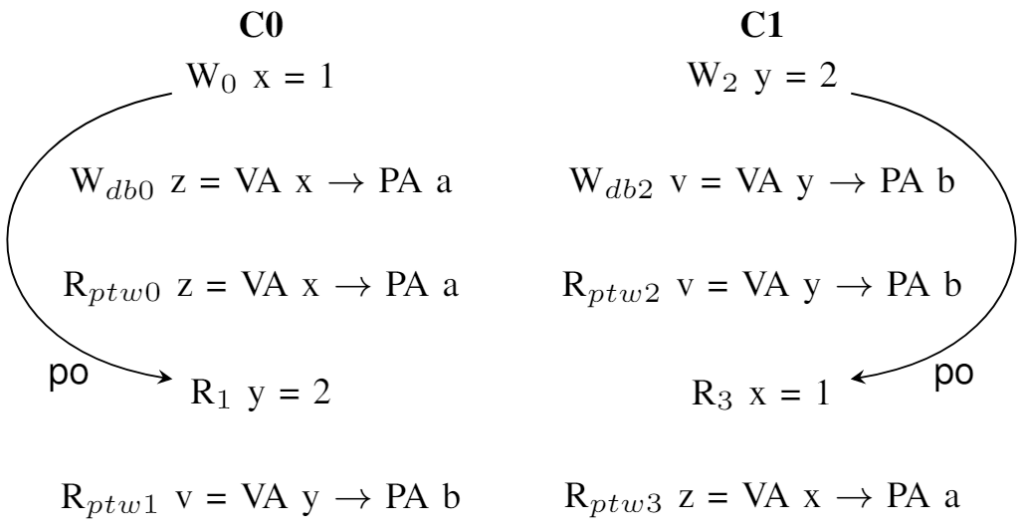
TransForm supports **page table walks (PT walks)** and **dirty bit updates**

Ghost instructions

PT walk: loads translation lookaside buffer (TLB) entry

dirty bit update: modifies dirty bit in page table entry (PTE)

ghost – relates user-facing MemoryEvent to invoked ghost instructions (numerical subscripts)



MTM Vocabulary: Hardware-level operations

TransForm supports **page table walks (PT walks)** and **dirty bit updates**

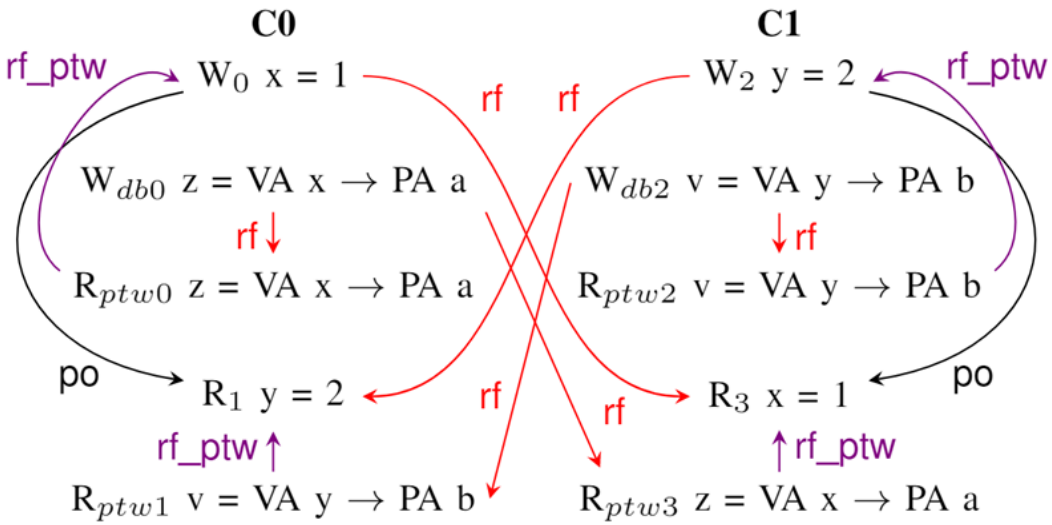
Ghost instructions

PT walk: loads translation lookaside buffer (TLB) entry

dirty bit update: modifies dirty bit in page table entry (PTE)

ghost – relates user-facing MemoryEvent to invoked ghost instructions (numerical subscripts)

rf_ptw – relates PT walk to user-facing MemoryEvents that access loaded TLB entry

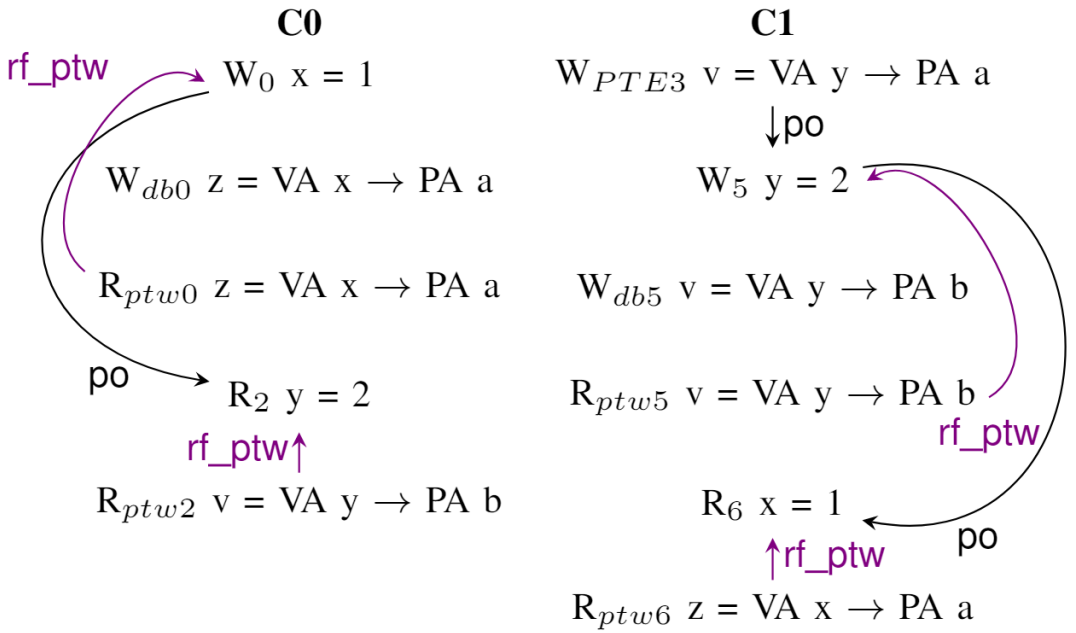


MTM Vocabulary: System-level operations

TransForm supports **address remappings via PTE Writes** and **TLB entry invalidations**

Support instructions

PTE Write: changes address mapping stored in a PTE for some VA v



MTM Vocabulary: System-level operations

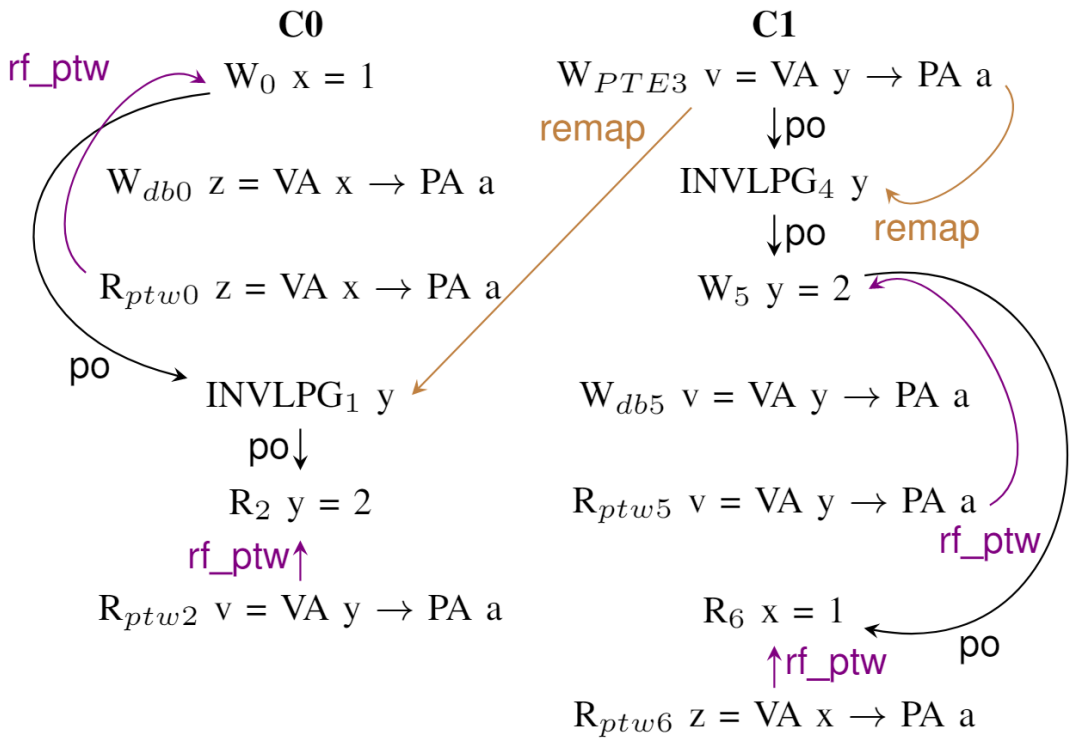
TransForm supports **address remappings via PTE Writes** and **TLB entry invalidations**

Support instructions

PTE Write: changes address mapping stored in a PTE for some VA v

INVLPG: invalidates TLB entry

remap – relates PTE Writes to invoked INVLPGs



MTM Vocabulary: System-level operations

TransForm supports **address remappings via PTE Writes** and **TLB entry invalidations**

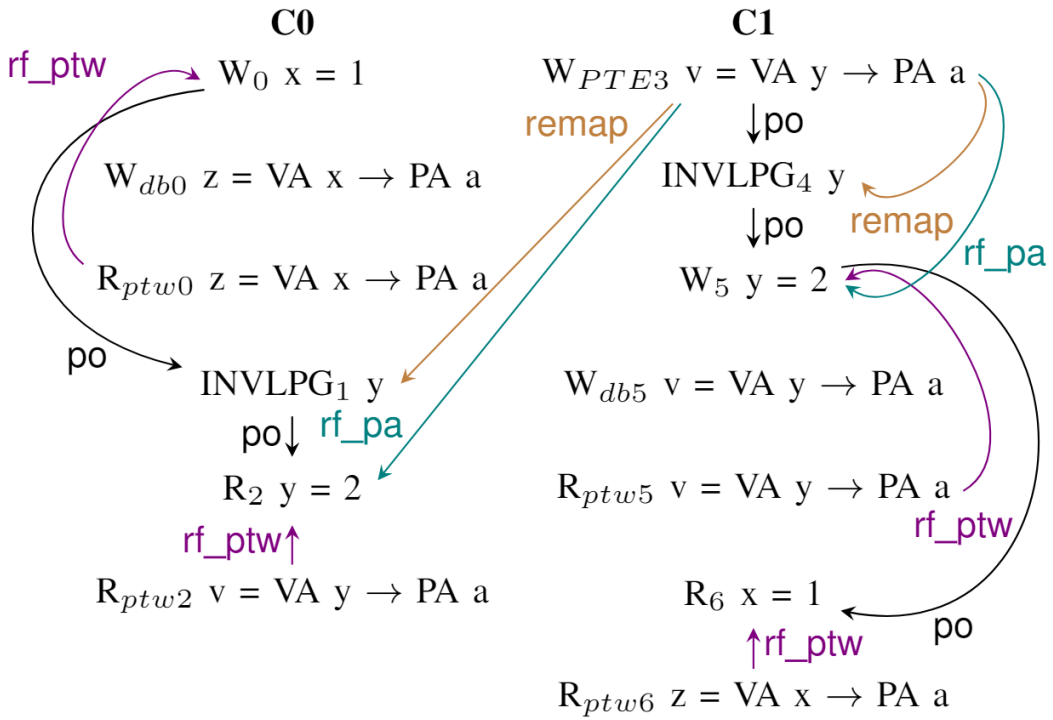
Support instructions

PTE Write: changes address mapping stored in a PTE for some VA v

INVLPG: invalidates TLB entry

remap – relates PTE Writes to invoked INVLPGs

rf_pa – relates PTE Write for VA v → PA p to user-facing MemoryEvents accessing PA p via VA v
co_pa, **fr_pa**, and **fr_va** follow similarly



MTM Vocabulary: System-level operations

TransForm supports **address remappings via PTE Writes** and **TLB entry invalidations**

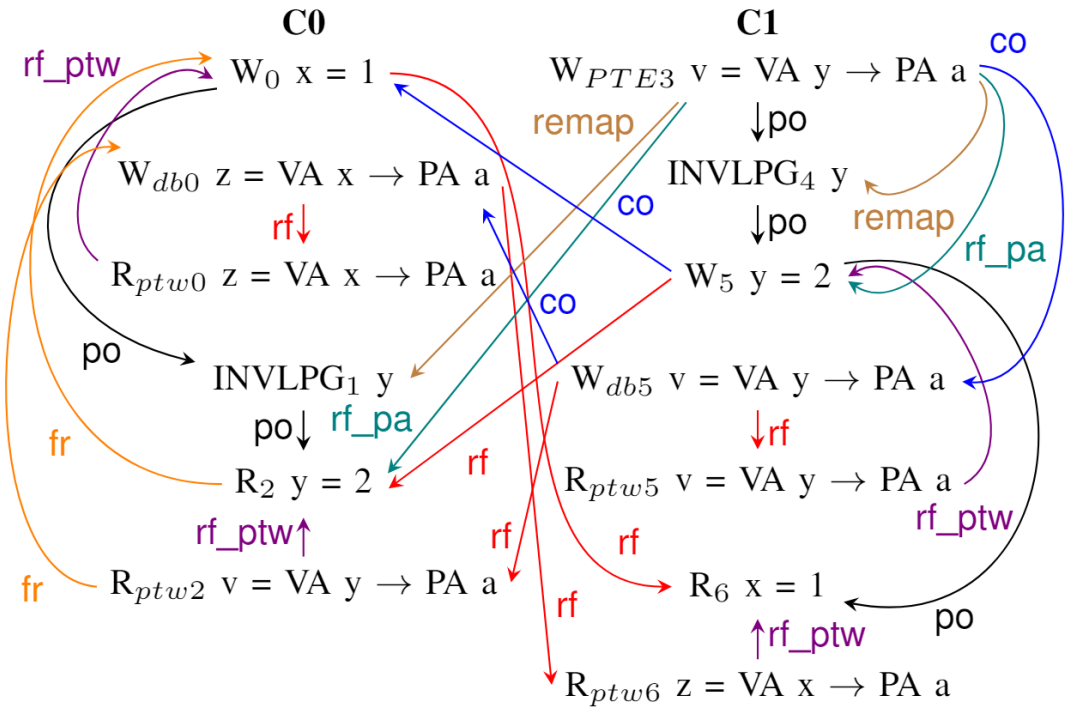
Support instructions

PTE Write: changes address mapping stored in a PTE for some VA v

INVLPG: invalidates TLB entry

remap – relates PTE Writes to invoked INVLPGs

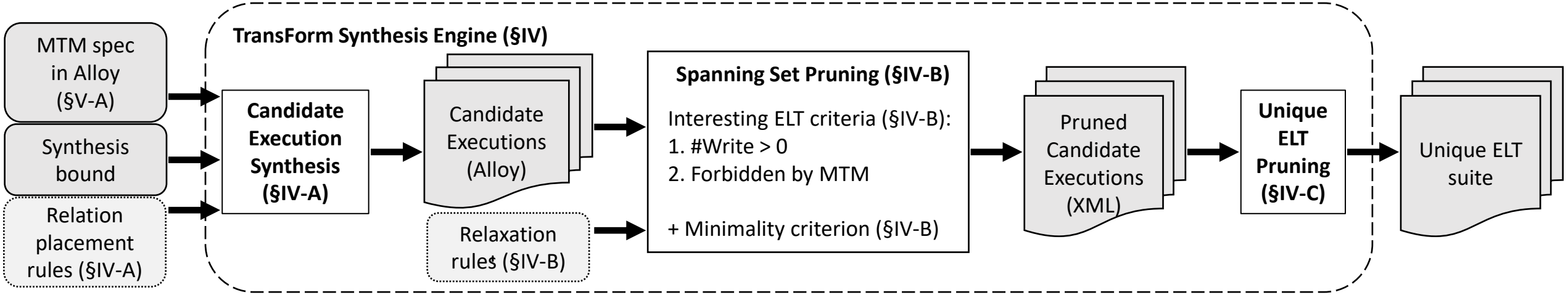
rf_pa – relates PTE Write for VA v → PA p to user-facing MemoryEvents accessing PA p via VA v
co_pa, **fr_pa**, and **fr_va** follow similarly



Outline

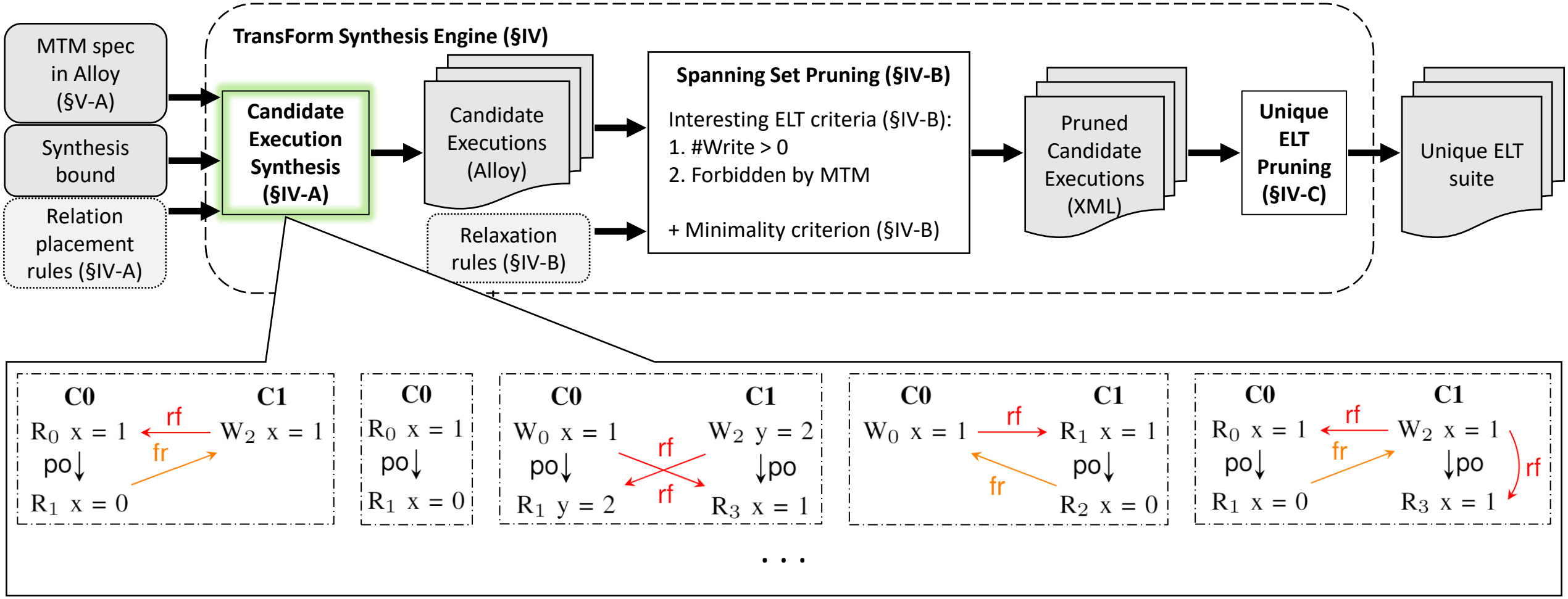
- Background on ISA-level MCM vocabulary
- Introduction to ISA-level MTM vocabulary
- Automating synthesis of ELTs
- Case Study: an estimated MTM for x86
- Conclusions

From Specification to Test Synthesis

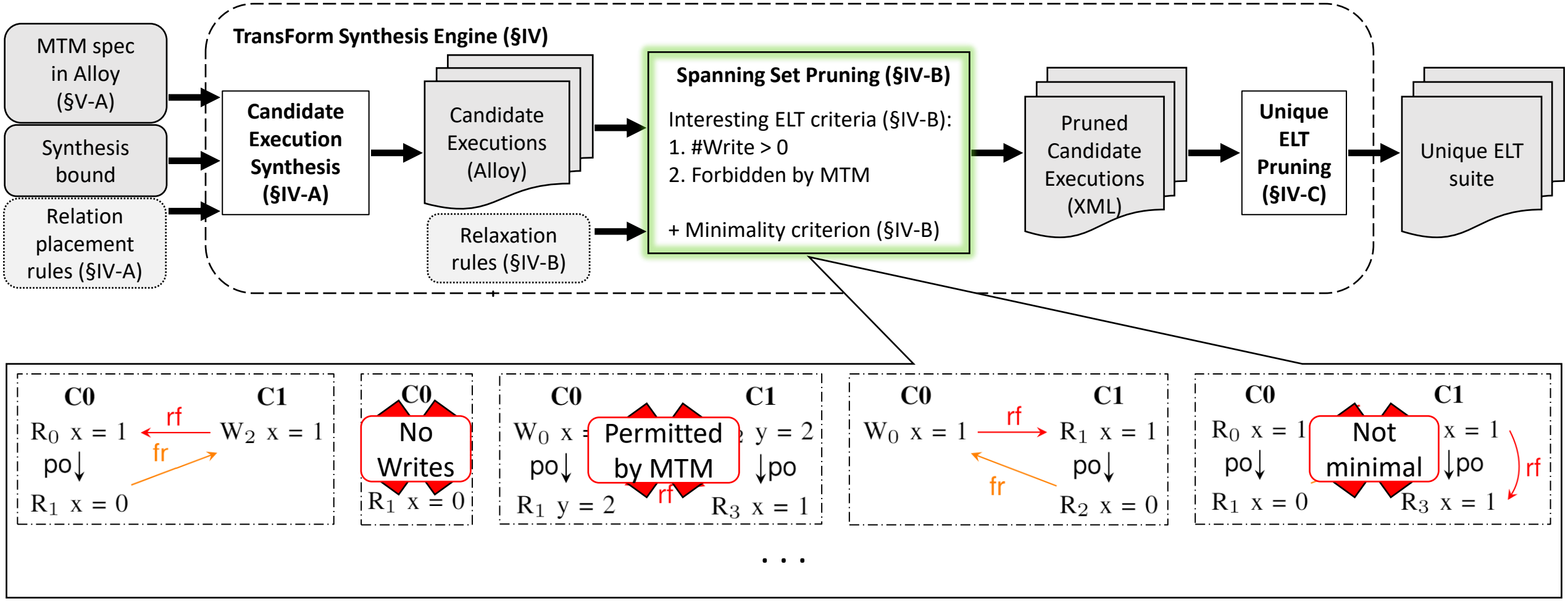


- ELTs include MTM vocabulary and support verification against an MTM spec
  - Goals:
    - Automated
    - Interesting and minimal (“Spanning set”)
    - Deduplicated
    - Comprehensive (to a bound)

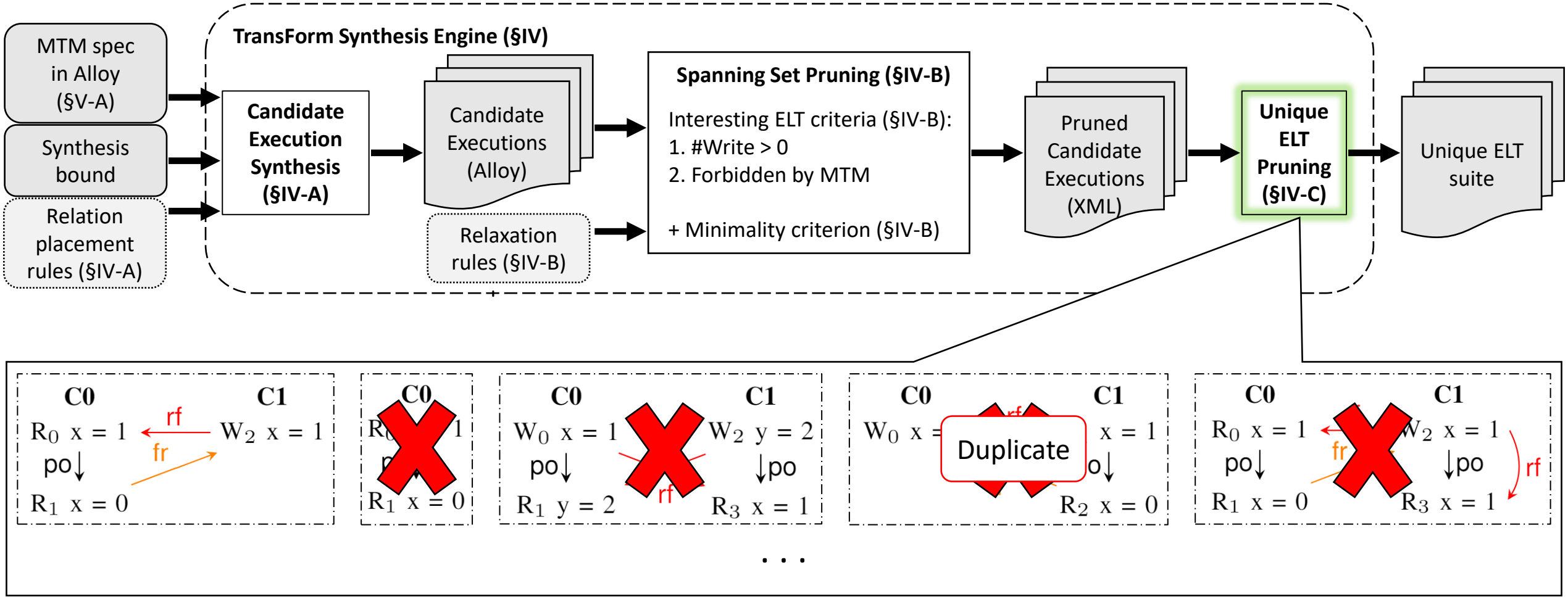
TransForm's synthesis engine starts by synthesizing all possible candidate executions up to a bound



Candidate executions are pruned for interesting ELT behaviors and checked for minimality



Unique ELTs are found by deduplicating synthesized ELTs with a post-processing script



Outline

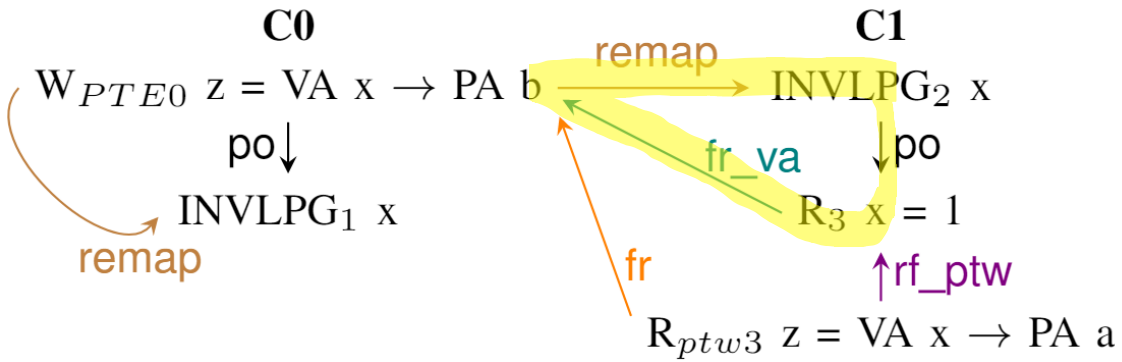
- Background on ISA-level MCM vocabulary
- Introduction to ISA-level MTM vocabulary
- Automating synthesis of ELTs
- Case Study: an estimated MTM for x86
- Conclusions

x86t_elt transistency predicates are composed of TSO axioms and new transistency-specific axioms

- **x86t_elt**: an approximate x86 transistency model based on prior work and publicly available documentation
- **x86-TSO**: sc_per_loc, rmw_atomicity, causality

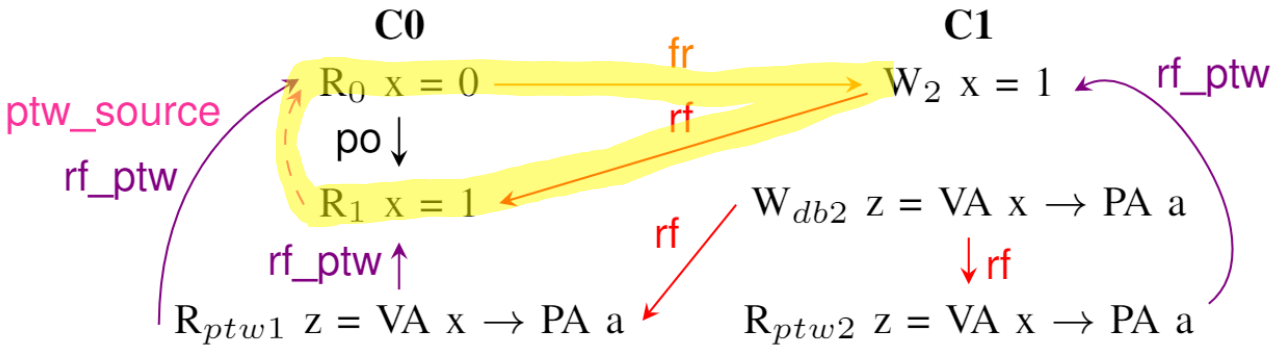
invlpg (required)

acylic[fr_va + remap + ^po]

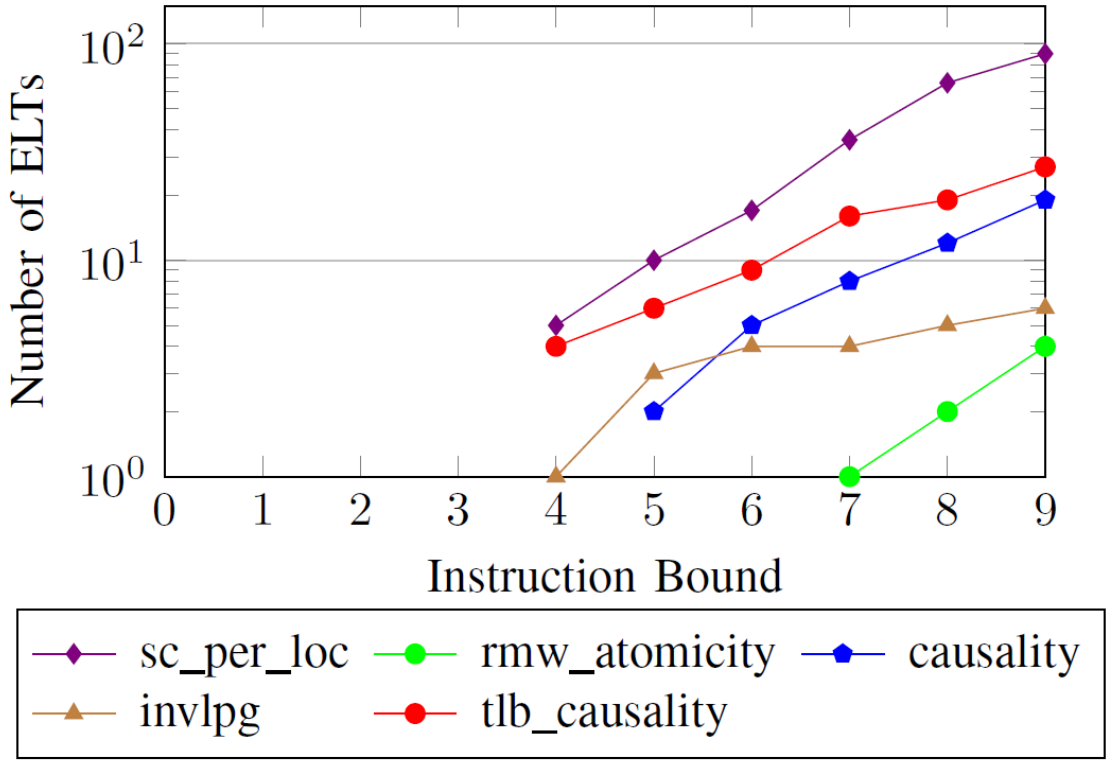


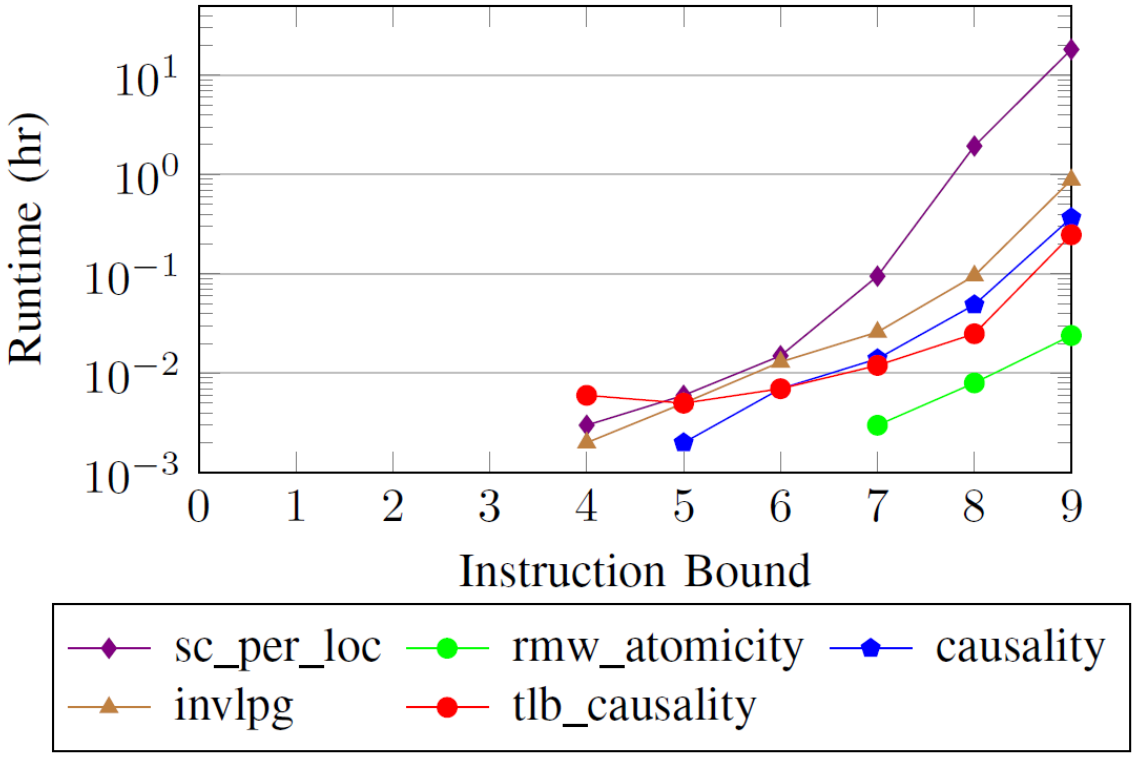
tlb_causality (auxiliary)

acyclic[ptw_source + com]



A per-axiom suite was synthesized for each **x86t_elt** axiom





103 total unique ELTs!

(98 for hardware verification/validation, 5 for diagnosing TLB implementation bugs)

The synthesized x86t_elt suite consisted of all relevant ELTs from prior work (up to the bound) and more

- 21 of 22 relevant ELTs from prior work synthesized
  - 6 ELTs synthesized verbatim → map to 3 ELT programs in x86t_elt suite
  - 15 ELTs can be reduced to a minimal ELT that is synthesized
  - 1 ELT requires a higher bound for synthesis
- 3 ELTs from prior work, 100 new ELTs

Conclusions

- **TransForm**: framework for formal specification of MTMs and ELT synthesis
- Enables modern ISAs to have a formal specification that includes VM
- Offers systems programmers and hardware designers a stronger opportunity for verification of full systems
- **Future work:**
  - Empirical x86 processor testing
  - RISC-V MTM specification
- Available at: https://github.com/naorinh/TransForm

TransForm: Formally Specifying Transistency Models and Synthesizing Enhanced Litmus Tests

Naorin Hossain
Princeton University

Caroline Trippel
Stanford University

Margaret Martonosi
Princeton University

ISCA 2020

https://github.com/naorinh/TransForm