Automatically Converting Axiomatic to Operational Models

Adwait Godbole[1] Yatin A. Manerkar[2] Sanjit A. Seshia[1]

[1]University of California, Berkeley [2]University of Michigan

FMCAD 2022

Axiomatic vs. Operational

Axiomatic models - **specification oriented**:

> “Behaviour of `counter`: it is incremented by one and resets at 15”

Operational models - **implementation oriented**:

```
counter : bv4

repeat(*)
    counter <= counter + 1;
```

Axiomatic vs. Operational

Axiomatic models **produce validity judgements over executions**:

```
Execution → { Valid, Invalid }
```

Operational models **produce valid executions**:

```
* → Execution
```
(`Valid` by definition)

Operational models are useful

- structural composition with system-under-test
- leveraging existing MC techniques
- compilation to substrates (hardware/emulation platforms)

Desire

Convert a given **axiomatic model** into an **equivalent operational model**

Desire

???

Convert a given **axiomatic model** into an **equivalent operational model**

???

???

Outline

- **Axiomatic microarchitectural models**
- Theoretical results
- Underapproximation and axiom automata
- Case studies

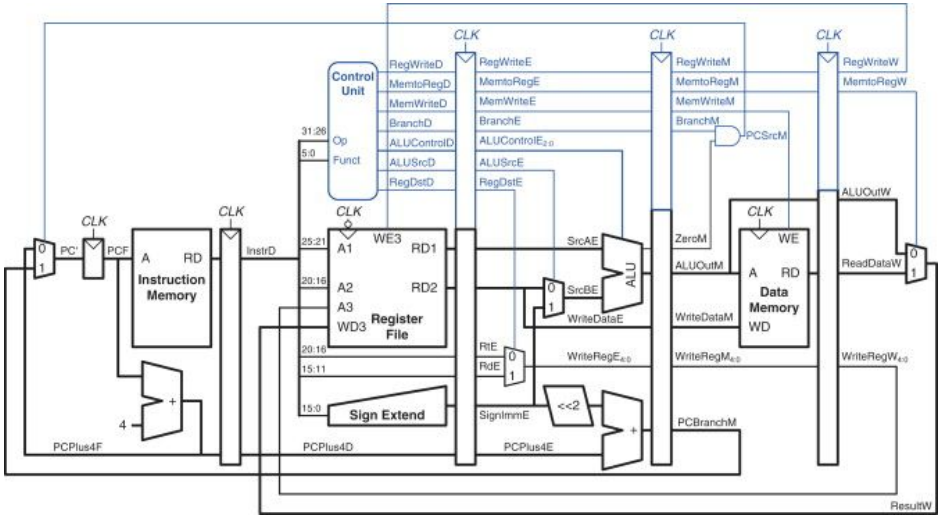
Microarchitectural models

Architectural execution

“net effect” of instructions

Microarchitectural execution

intra-instruction detail

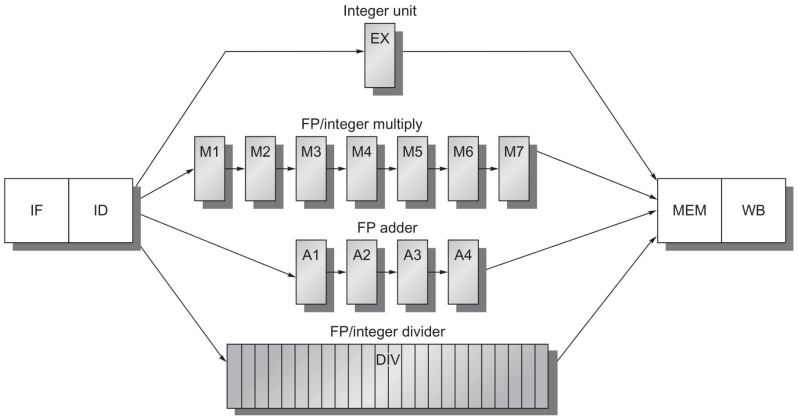


Microarchitectural execution proceeds in stages

μspec: *Axiomatic* microarchitectural models

Modelling feature:

Decompose instruction execution into *events*



Axioms allow specification of orderings between events:

"FP add will reach A4 before previous FP multiply reaches M7"

μspec: a DSL for microarchitectural ordering models

Formalizes the syntax and semantics of ordering axioms

$$\forall\ \texttt{i1, i2, } (\texttt{i1} <_r \texttt{i2} \wedge \texttt{DepOn(i1, i2)}) \implies \texttt{hb(i1.Exe, i2.Exe)}$$

Lustig et al. “COATCheck: Verifying Memory Ordering at the Hardware-OS Interface” [ASPLOS 2016]

μspec: a DSL for microarchitectural ordering models

Formalizes the syntax and semantics of ordering axioms

quantification over instructions

$$\forall\ \texttt{i1, i2},\ (\texttt{i1} <_r \texttt{i2} \wedge \texttt{DepOn(i1, i2)}) \implies \texttt{hb(i1.Exe, i2.Exe)}$$

Lustig et al. “COATCheck: Verifying Memory Ordering at the Hardware-OS Interface” [ASPLOS 2016]

μspec: a DSL for microarchitectural ordering models

Formalizes the syntax and semantics of ordering axioms

quantification over instructions

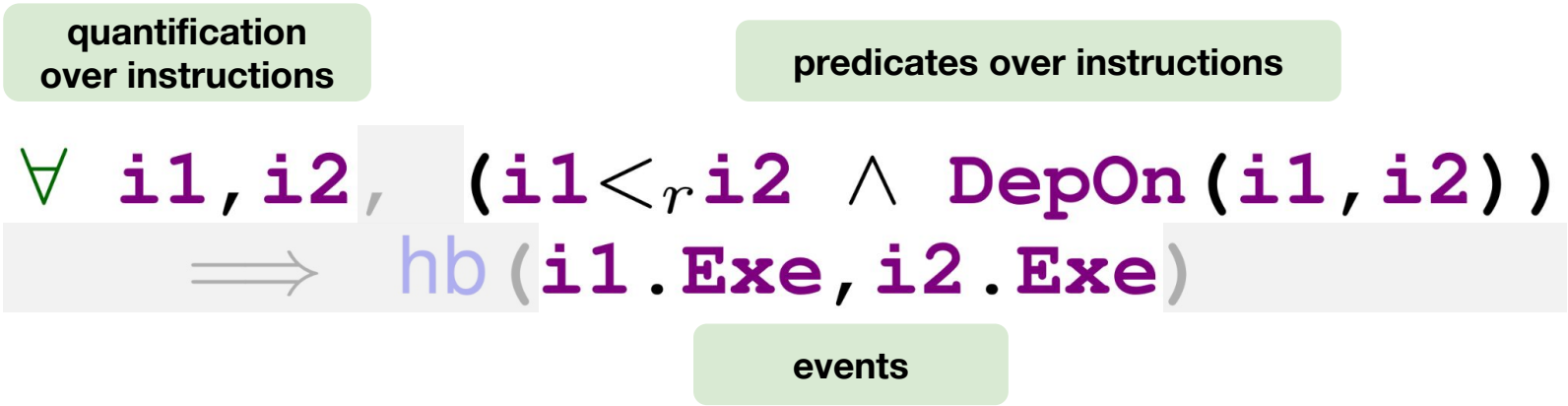
$$\forall\ \texttt{i1}, \texttt{i2},\ (\texttt{i1} <_r \texttt{i2} \wedge \texttt{DepOn(i1,i2)}) \implies \texttt{hb(i1.Exe, i2.Exe)}$$

events

Lustig et al. “COATCheck: Verifying Memory Ordering at the Hardware-OS Interface” [ASPLOS 2016]

μspec: a DSL for microarchitectural ordering models

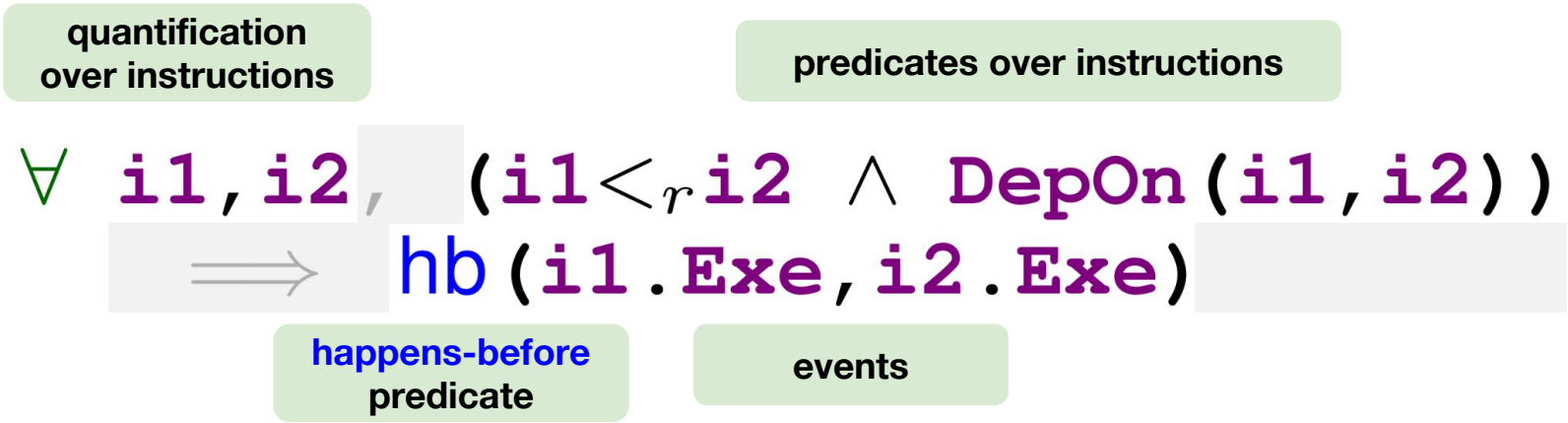
Formalizes the syntax and semantics of ordering axioms



Lustig et al. “COATCheck: Verifying Memory Ordering at the Hardware-OS Interface” [ASPLOS 2016]

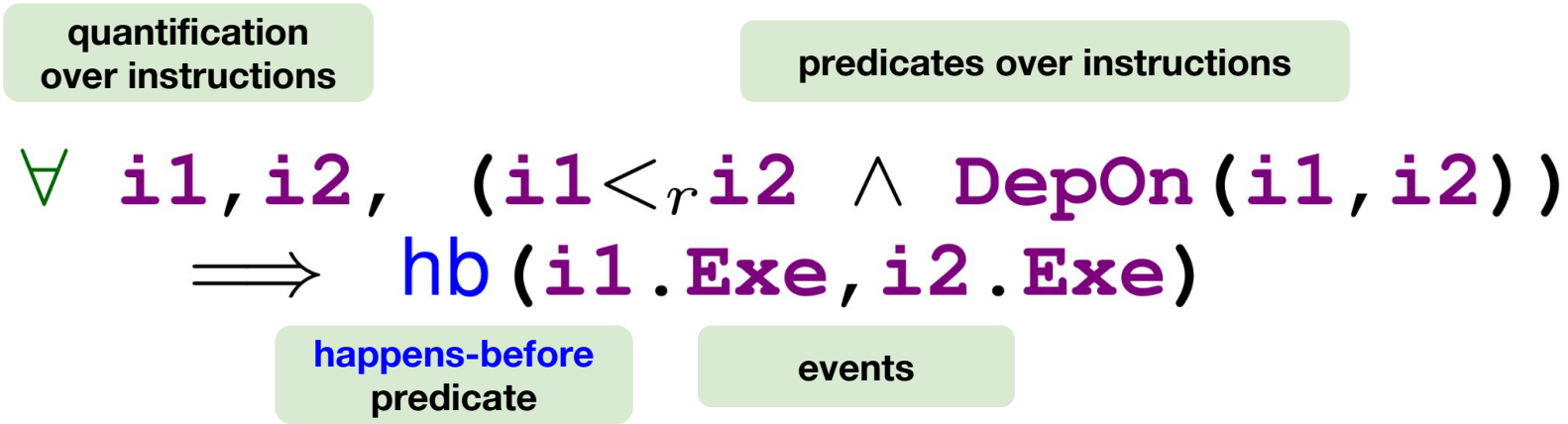
μspec: a DSL for microarchitectural ordering models

Formalizes the syntax and semantics of ordering axioms



μspec: a DSL for microarchitectural ordering models

Formalizes the syntax and semantics of ordering axioms



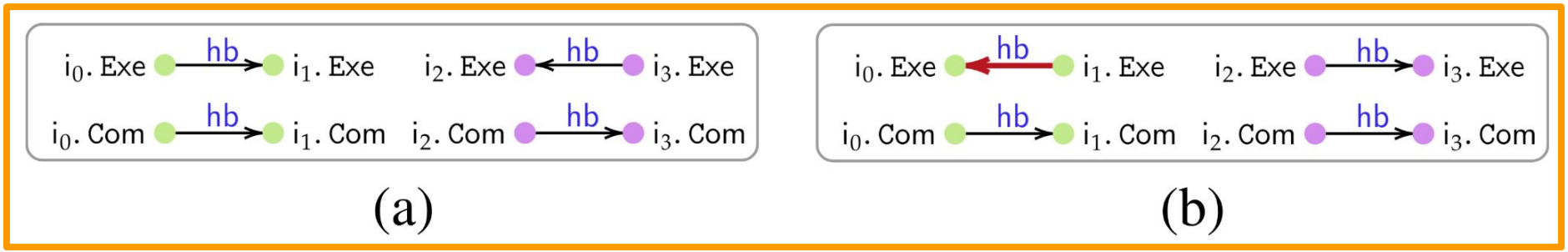
Lustig et al. “COATCheck: Verifying Memory Ordering at the Hardware-OS Interface” [ASPLOS 2016]

Axioms are interpreted over graphs

$G \models_{\mathcal{P}} \mathrm{AX}$

$\forall$ i1, i2, (i1 $<_r$ i2 $\wedge$ DepOn(i1, i2)) $\implies$ hb(i1.Exe, i2.Exe)

i_0: lw r1, 42(r0)	i_2: lw r4, 42(r0)
i_1: add r3, r2, r1	i_3: add r3, r2, r1



Choice of axiomatic framework: μspec

μspec has been used widely

- memory consistency — **PipeCheck [MICRO 14]**
- verification against RTL — **RTLCheck [MICRO 17]**
- cache coherence — **CCICheck [MICRO 15]**
- security analysis — **CheckMate [MICRO 19]**

has semantic resemblance with event structure-like models

Manual axiomatic ⟷ operational equivalence

An Operational Semantics for C/C++11 Concurrency

Kyndylan Nienhuis Kayvan Memarian Peter Sewell
University of Cambridge
United Kingdom
first.last@cl.cam.ac.uk

A Better x86 Memory Model: x86-TSO

Scott Owens Susmit Sarkar Peter Sewell
University of Cambridge
http://www.cl.cam.ac.uk/users/pes20/weakmemory

Taming Release-Acquire Consistency

Ori Lahav Nick Giannarakis Viktor Vafeiadis
Max Planck Institute for Software Systems (MPI-SWS), Germany

Problem statement

μspec

Convert a given **axiomatic model** into an **equivalent operational model**

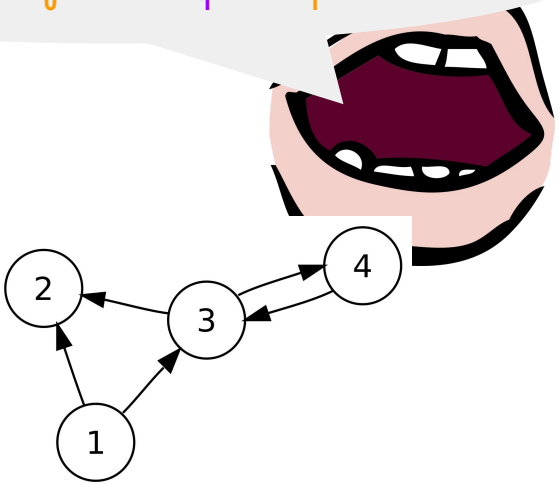
???

???

Operational model

totally ordered sequence of events:

i_0.Fet i_0.Fet i_0.Exe … i_1.Exe i_1.Exe



Formally:

Multi-input, single-output transducer

i_0	i_1	…	…	
i_0	i_1	…	…	

Operational model

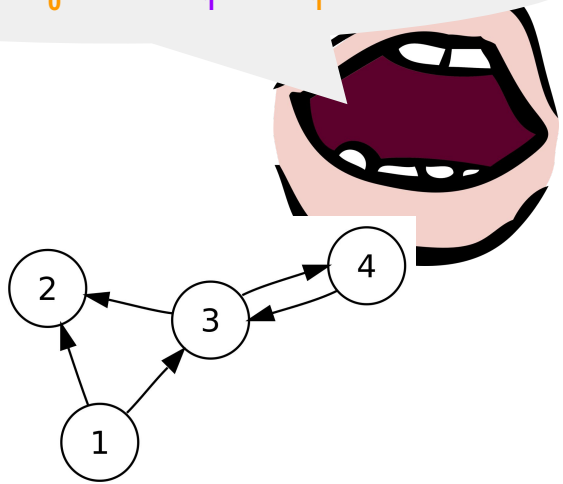
totally ordered sequence of events:

i_0.Fet i_0.Fet i_0.Exe … i_1.Exe i_1.Exe

transition (outputs) labelled with events

control state

input tapes with instructions



i_0	i_1	…	…	
i_0	i_1	…	…	

Operational model

Desideratum: **finite state**

totally ordered sequence of events:

i_0.Fet i_0.Fet i_0.Exe … i_1.Exe i_1.Exe

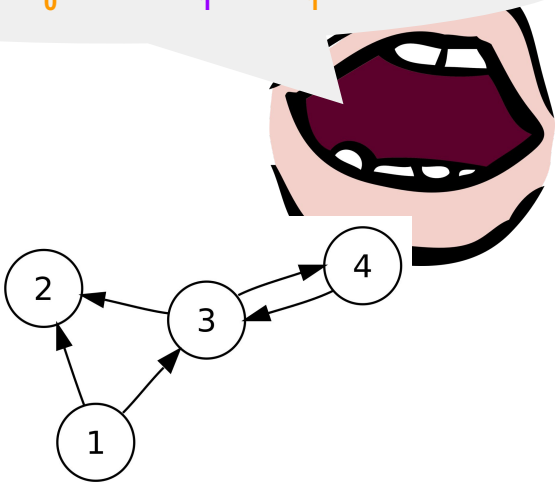
transition (outputs) labelled with events

finite control state

control state

bounded history of instructions

input tapes with instructions



i_0	i_1	…	…	
i_0	i_1	…	…	

Outline

- Axiomatic microarchitectural models
- **Theoretical results**
- Underapproximation and axiom automata
- Experimental results

Problem statement

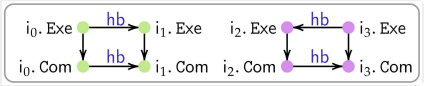
μspec

Convert a given **axiomatic model** into an **equivalent operational model**

???

transducer-like model

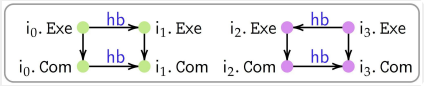
Alignment issues



i_0.Fet i_0.Fet i_0.Exe … i_1.Exe i_1.Exe

Axiomatic model	Operational model
`Execution → { Valid, Invalid }`	`* → Execution`
Partially ordered executions	Totally ordered executions
Validity only over complete executions	Incremental generation of execution requires validity judgement at each step

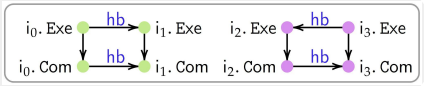
Alignment issues



i_0.Fet i_0.Fet i_0.Exe … i_1.Exe i_1.Exe

Axiomatic model	**Operational model**
`Execution → { Valid, Invalid }`	`* → Execution`
Partially ordered executions	Totally ordered executions
(consider linearizations)	
Validity only over complete executions	Incremental generation of execution requires validity judgment at each step
(local liveness guarantee)	

Alignment issues



i_0.Fet i_0.Fet i_0.Exe … i_1.Exe i_1.Exe

Axiomatic model	**Operational model**
`Execution → { Valid, Invalid }`	`* → Execution`
Partially ordered executions	Totally ordered executions
Validity only over complete executions	Incremental generation of execution requires validity ment at each step

refinability

extensibility

Refinability and extensibility

Refinability:

Linearizations of valid graphs are valid

Extensibility:

Partial executions can be stitched together to form valid complete executions

Refinability and extensibility

Refinability: **efficiently checkable**

Linearizations of valid graphs are valid

Extensibility: **efficiently checkable**

Partial executions can be stitched together to form valid complete executions

μspecRE: subset of μspec which has refinable and extensible axioms

Equivalence

Operational model	Axiomatic model
Soundness	
Every generated execution…	should be a linearization of a valid graph
Completeness	
All linearizations should be generated…	for every valid graph.

Impossibility result

Thm 1. Axiomatic to (finite) operational models conversion is not generally possible

$$\texttt{ax0}: \forall\ \texttt{i1} \qquad \mathrm{hb}(\texttt{i1.S}, \texttt{i1.T})$$
$$\texttt{ax1}: \forall\ \texttt{i1}, \texttt{i2} \quad \mathrm{hb}(\texttt{i1.S}, \texttt{i2.S}) \implies \mathrm{hb}(\texttt{i1.T}, \texttt{i2.T})$$

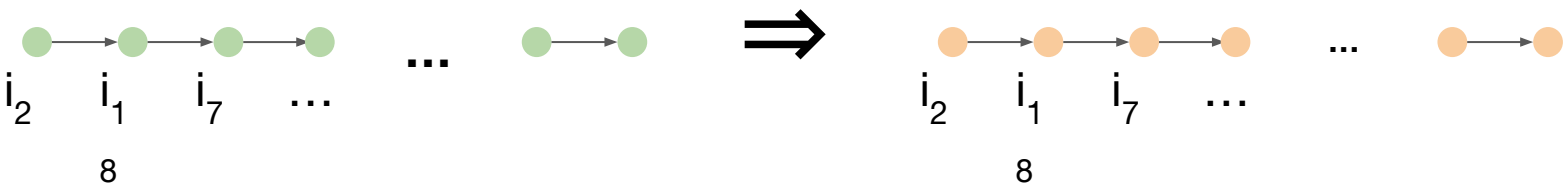
Axiom $A^{\#}$: “order of T-events should be identical to order of S-events”

Impossibility result

Thm 1. Axiomatic to (finite) operational models conversion is not generally possible

$$\texttt{ax0}: \forall\ \texttt{i1} \quad \mathrm{hb}(\texttt{i1.S}, \texttt{i1.T})$$
$$\texttt{ax1}: \forall\ \texttt{i1}, \texttt{i2} \quad \mathrm{hb}(\texttt{i1.S}, \texttt{i2.S}) \implies \mathrm{hb}(\texttt{i1.T}, \texttt{i2.T})$$

Axiom $A^{\#}$: “order of T-events should be identical to order of S-events”



Outline

- Axiomatic microarchitectural models
- Theoretical results
- **Underapproximation and axiom automata**
- Case studies

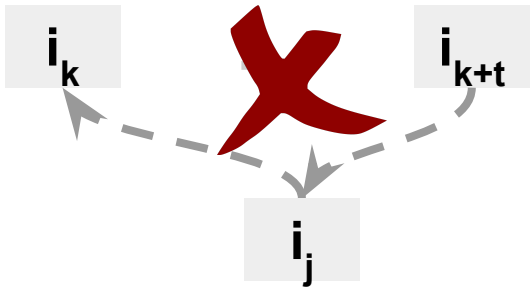
Bounded reorderings

Relax completeness to **t-bounded completeness**

t-bounded completeness:

- Reordering depth is limited to **t** instructions

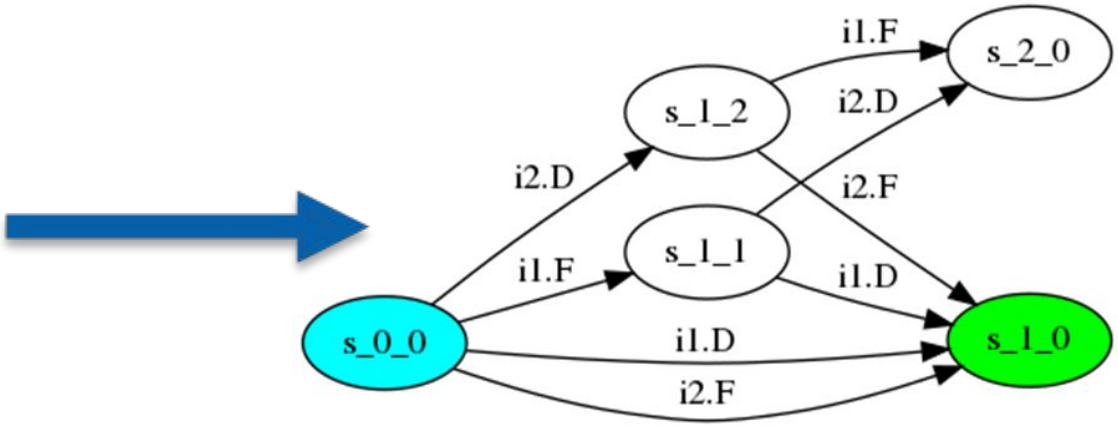
- A core cannot be starved while > **t** instructions execute on another



Axiom automata

Monitor for axiom violation

$$\forall\ \texttt{i1,i2.}\ \ \mathrm{hb}(\texttt{i1.F,i1.D}) \implies \mathrm{hb}(\texttt{i2.F,i2.D})$$



Key result: only bounded-many automata for t-bounded executions

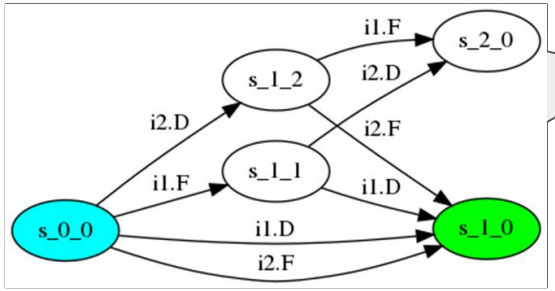
⇒ **Thm 2. *finite state* operationalization is possible for t-bounded executions**

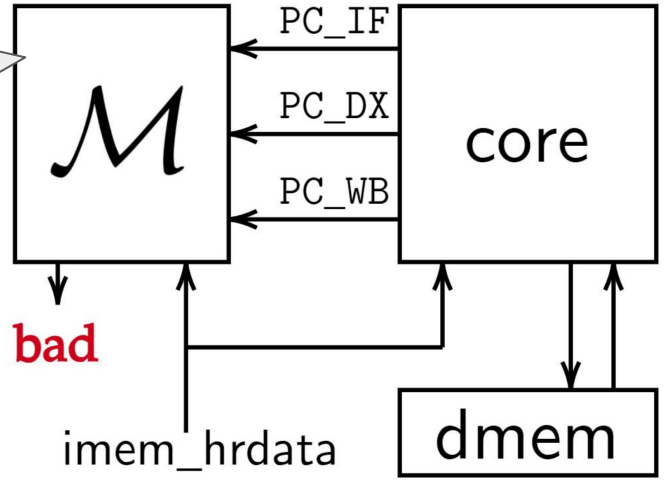
Outline

- Axiomatic microarchitectural models
- Theoretical results
- Under-approximation and axiom automata
- **Case studies**

Case studies

1. **Applying symbolic TS techniques (e.g. PDR) for axiomatic models**





Instructions	PDR	BMC ($d = 20$)
ALU-I	1m46s	11m30s
ALU-R	1m56s	15m40s
Load+Store	2m20s	13m37s

Case studies

2. Technique is not limited to processor pipelines

Verification of host interface for SDRAM controller

writes, reads and SDRAM bank refresh

3. Also can help in bug hunting

RAW dependency violation in OoO processor due to incorrect RAT reset

$$\texttt{axDep}: \forall\ \texttt{i1},\ \texttt{i2},\ (\texttt{i1} <_r \texttt{i2} \wedge \texttt{Cons(i1,i2)} \wedge \texttt{DepOn(i1,i2)}) \Rightarrow \texttt{hb(i1.E,\ i2.E)}$$

Future work

1. Static checks (e.g. type systems) for μspecRE
2. Quantitative extensions to μspec
3. Richer operational models

Conclusion

Contributions at a glance:

Conclusion

Contributions at a glance:

- Study of the alignment problem between axiomatic and operational models
  Refinability and extensibility

Conclusion

Contributions at a glance:

- Study of the alignment problem between axiomatic and operational models
  Refinability and extensibility
- General μspec makes for a costly verification problem
  Finite operationalization is not possible unconditionally

Conclusion

Contributions at a glance:

- Study of the alignment problem between axiomatic and operational models
  Refinability and extensibility
- General μspec makes for a costly verification problem
  Finite operationalization is not possible unconditionally
- Underapproximations are possible and useful
  t-bounded underapproximation allows finite operationalization

Conclusion

Contributions at a glance:

- Study of the alignment problem between axiomatic and operational models
  Refinability and extensibility
- General μspec makes for a costly verification problem
  Finite operationalization is not possible unconditionally
- Underapproximations are possible and useful
  t-bounded underapproximation allows finite operationalization
- Operationalization has verification value
  More easily connects with HW design
  Opens the door to TS-based model checking

END