A FORMAL ANALYSIS OF THE NVIDIA PTX MEMORY CONSISTENCY MODEL

Dan Lustig, Sameer Sahasrabuddhe, Olivier Giroux, Apr 15, 2019 (ASPLOS 2019)

THE NVIDIA PTX MEMORY CONSISTENCY MODEL

Memory consistency model: a set of rules defining the values that loads can legally return

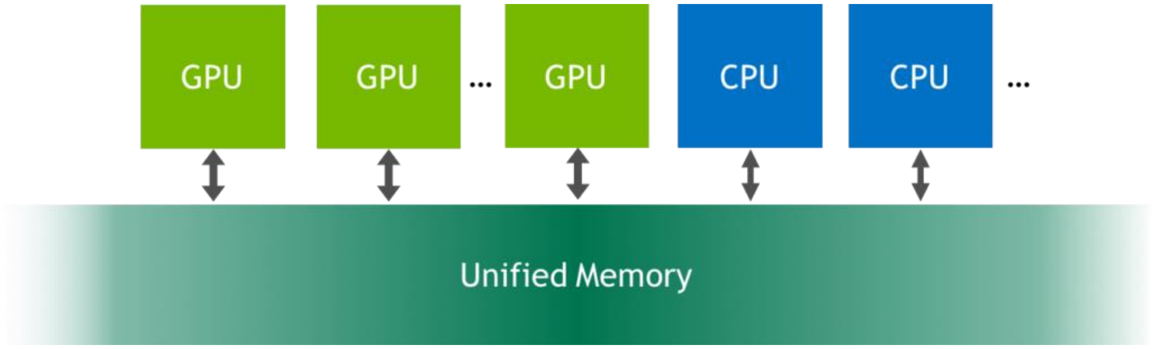


Image: https://devblogs.nvidia.com/unified-memory-cuda-beginners/

This paper: a new scoped memory consistency model for NVIDIA GPUs

- enables flexible intra-GPU, GPU-GPU, and CPU-GPU communication
- PTX documentation online, and axiomatic formalization in paper
- C++ compiler mappings tested using Alloy and verified using Coq

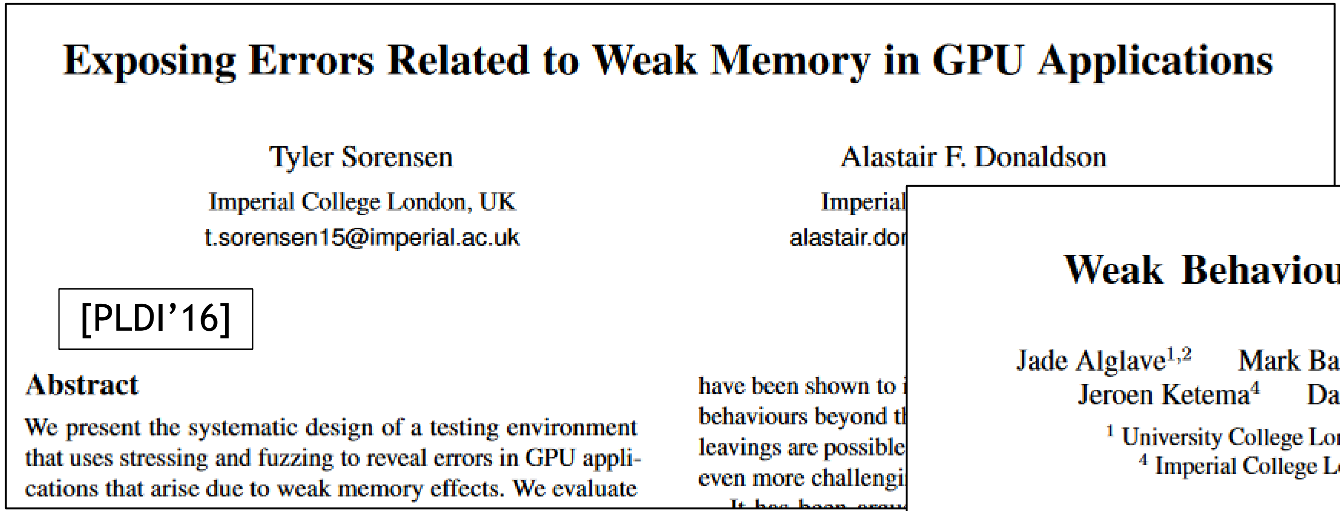
Lustig, Sahasrabuddhe, Giroux, “A Formal Analysis of the NVIDIA PTX Memory Consistency Model”, ASPLOS 2019

MOTIVATION

Why create a new memory consistency model for NVIDIA GPUs?

GOAL #1: FIX BUGS IN PRIOR GENERATIONS

Implementation (compiler + hardware) must respect software specs



Exposing Errors Related to Weak Memory in GPU Applications

Tyler Sorensen
Imperial College London, UK
t.sorensen15@imperial.ac.uk

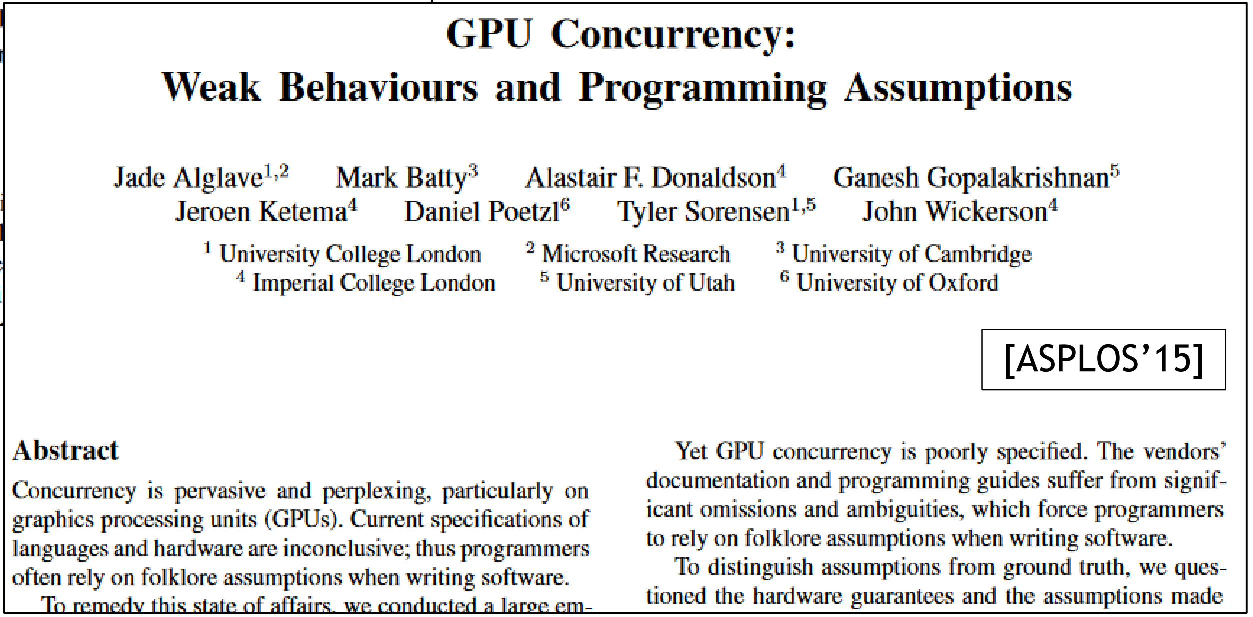
Alastair F. Donaldson
Imperial
alastair.dor

[PLDI'16]

Abstract

We present the systematic design of a testing environment that uses stressing and fuzzing to reveal errors in GPU applications that arise due to weak memory effects. We evaluate

have been shown to i
behaviours beyond th
leavings are possible
even more challengi



GPU Concurrency:
Weak Behaviours and Programming Assumptions

Jade Alglave[1,2] Mark Batty[3] Alastair F. Donaldson[4] Ganesh Gopalakrishnan[5]
Jeroen Ketema[4] Daniel Poetzl[6] Tyler Sorensen[1,5] John Wickerson[4]
[1] University College London [2] Microsoft Research [3] University of Cambridge
[4] Imperial College London [5] University of Utah [6] University of Oxford

[ASPLOS'15]

Abstract

Concurrency is pervasive and perplexing, particularly on graphics processing units (GPUs). Current specifications of languages and hardware are inconclusive; thus programmers often rely on folklore assumptions when writing software.

To remedy this state of affairs, we conducted a large em-

Yet GPU concurrency is poorly specified. The vendors' documentation and programming guides suffer from significant omissions and ambiguities, which force programmers to rely on folklore assumptions when writing software.

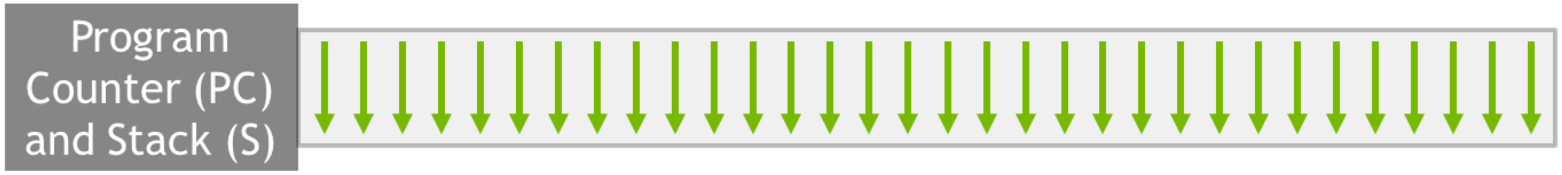
To distinguish assumptions from ground truth, we questioned the hardware guarantees and the assumptions made

Lustig, Sahasrabuddhe, Giroux, "A Formal Analysis of the NVIDIA PTX Memory Consistency Model", ASPLOS 2019

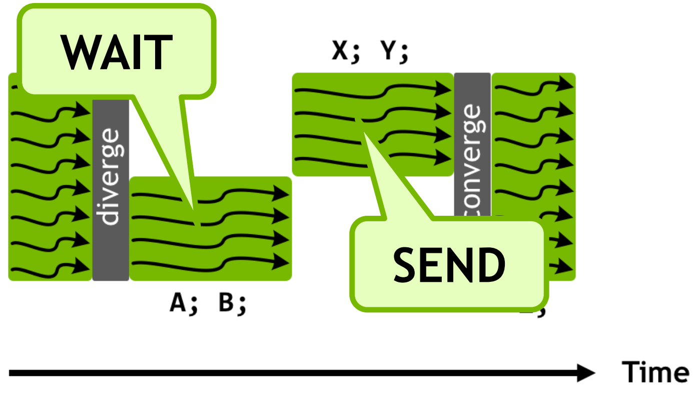
GOAL #2: ENABLE AN IMPROVED SIMT MODEL

Avoid starvation/livelock scenarios possible under prior SIMT model

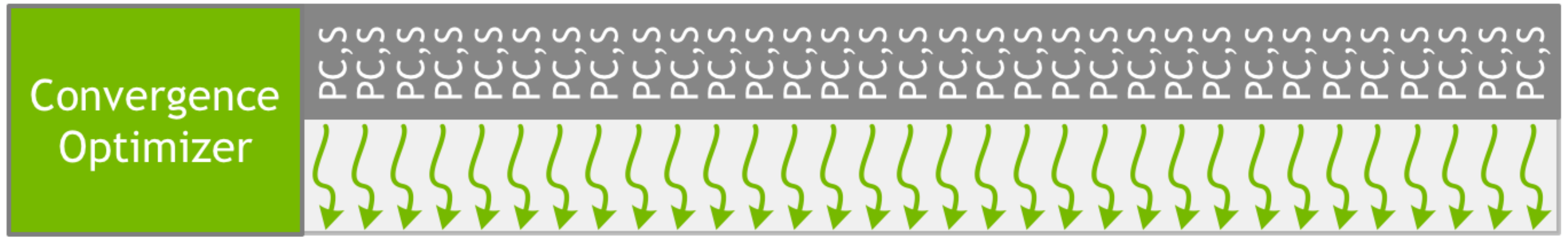
Pre-Volta



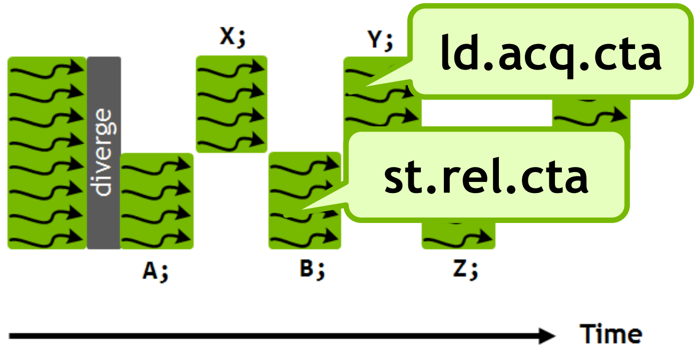
32 thread warp



Volta



32 thread warp with independent scheduling



https://devblogs.nvidia.com/inside-volta/

THE NVIDIA MEMORY CONSISTENCY MODEL

A High-Level Overview

NVIDIA MEMORY MODEL: GENERAL APPROACH

1. Start with the least common denominator of modern CPU weak memory consistency models
   - As weak as possible (better performance, more microarchitectural flexibility), as long as the result is properly programmable
2. Add scopes

PTX MEMORY MODEL STARTING POINT

The “Least Common Denominator” of existing models

- **Causality**
  - Release/Acquire, CTA Execution Barriers, Fences (transitively)
- **Coherence Order**
- **Sequential Consistency per Location**
  - expected single-threaded and same-address behavior
- **Atomicity of RMWs**
- **No Out-of-Thin-Air Executions**
  - still-open theoretical problem: how to prevent self-justifying speculation?

Lustig, Sahasrabuddhe, Giroux, “A Formal Analysis of the NVIDIA PTX Memory Consistency Model”, ASPLOS 2019

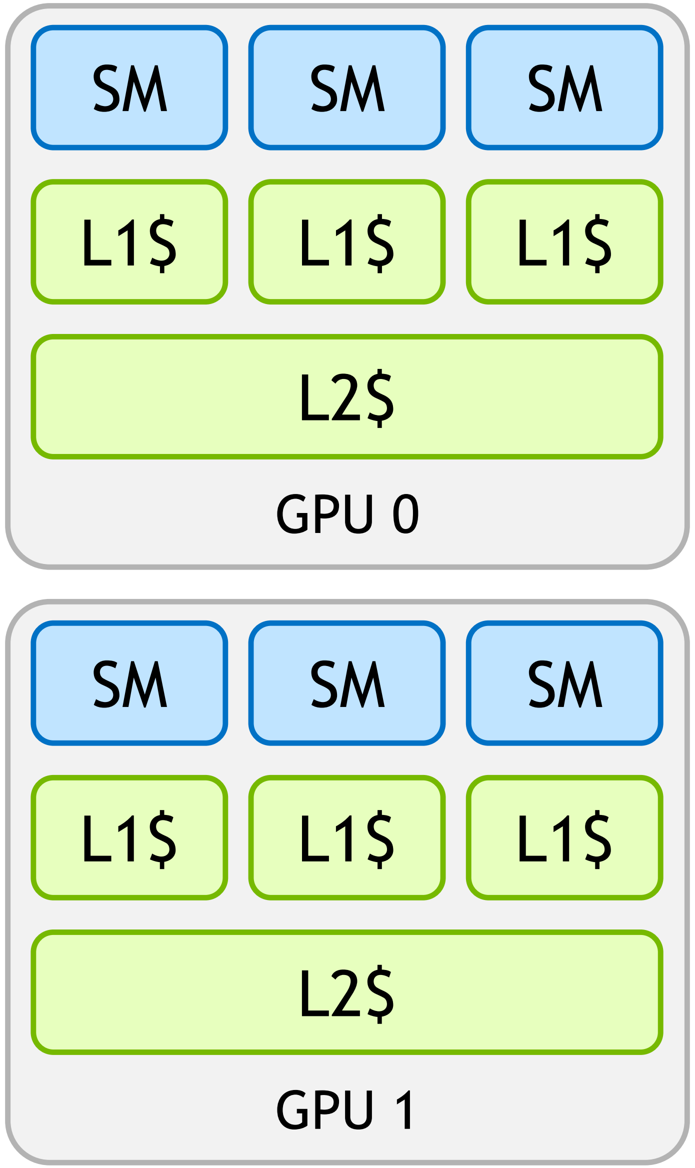
NVIDIA MEMORY MODEL: GENERAL APPROACH

1. Start with the least common denominator of modern CPU weak memory consistency models
   - As weak as possible (better performance, more microarchitectural flexibility), as long as the result is properly programmable

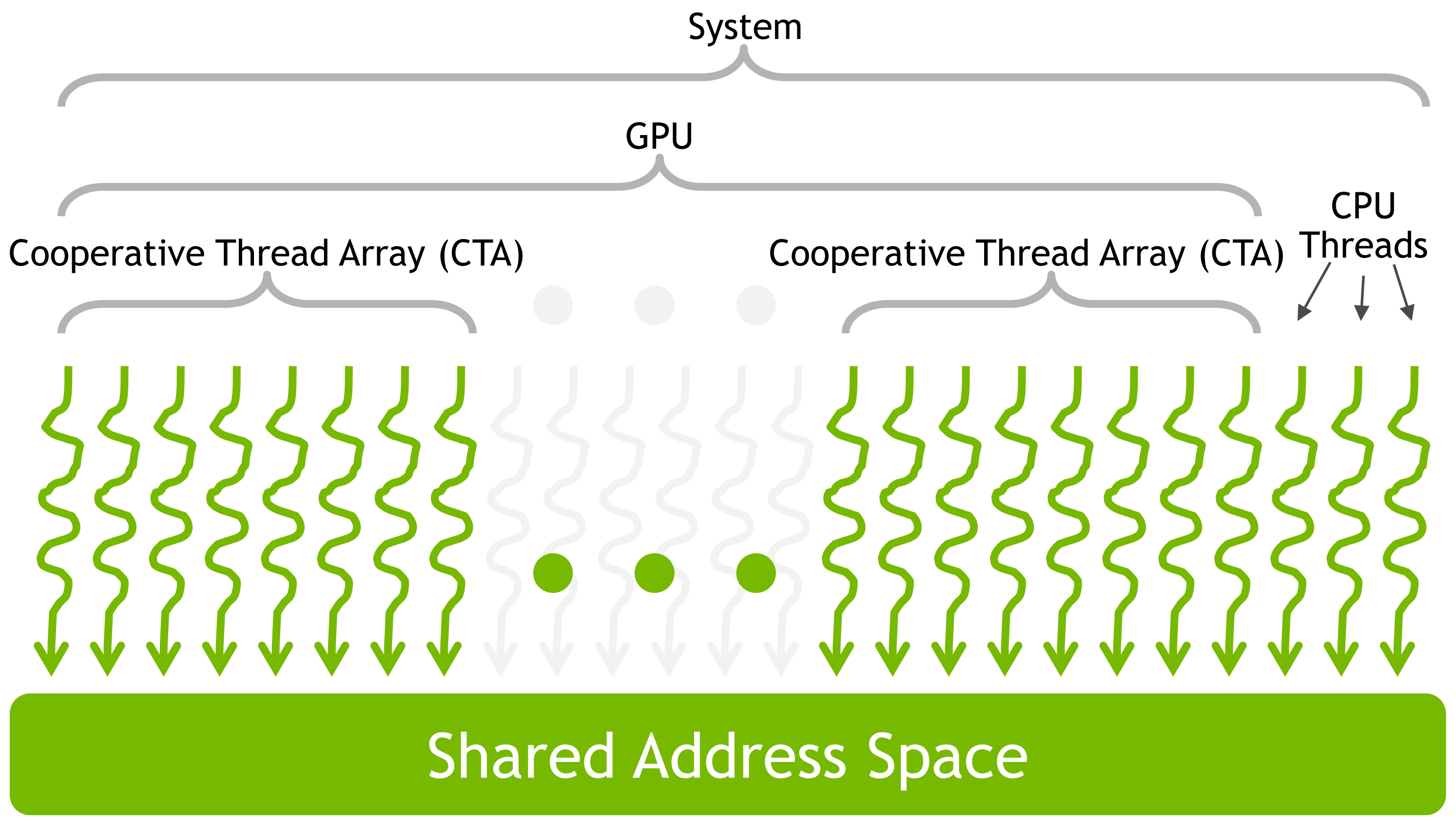
2. Add scopes

SCOPES OR NO SCOPES?

- Early GPU memory models exposed scopes
  - HSA/HRF [ASPLOS'14], OpenCL
- Later papers: HW coherence protocols can track scopes instead of user, delivering simpler model with no performance loss
  - Remote Scope Promotion [ASPLOS'15], DeNovo+DRF [MICRO'15], Relativistic Cache coherence [HPCA'17], …
- But: that required extra hardware support is a non-trivial cost we can't/won't commit to!

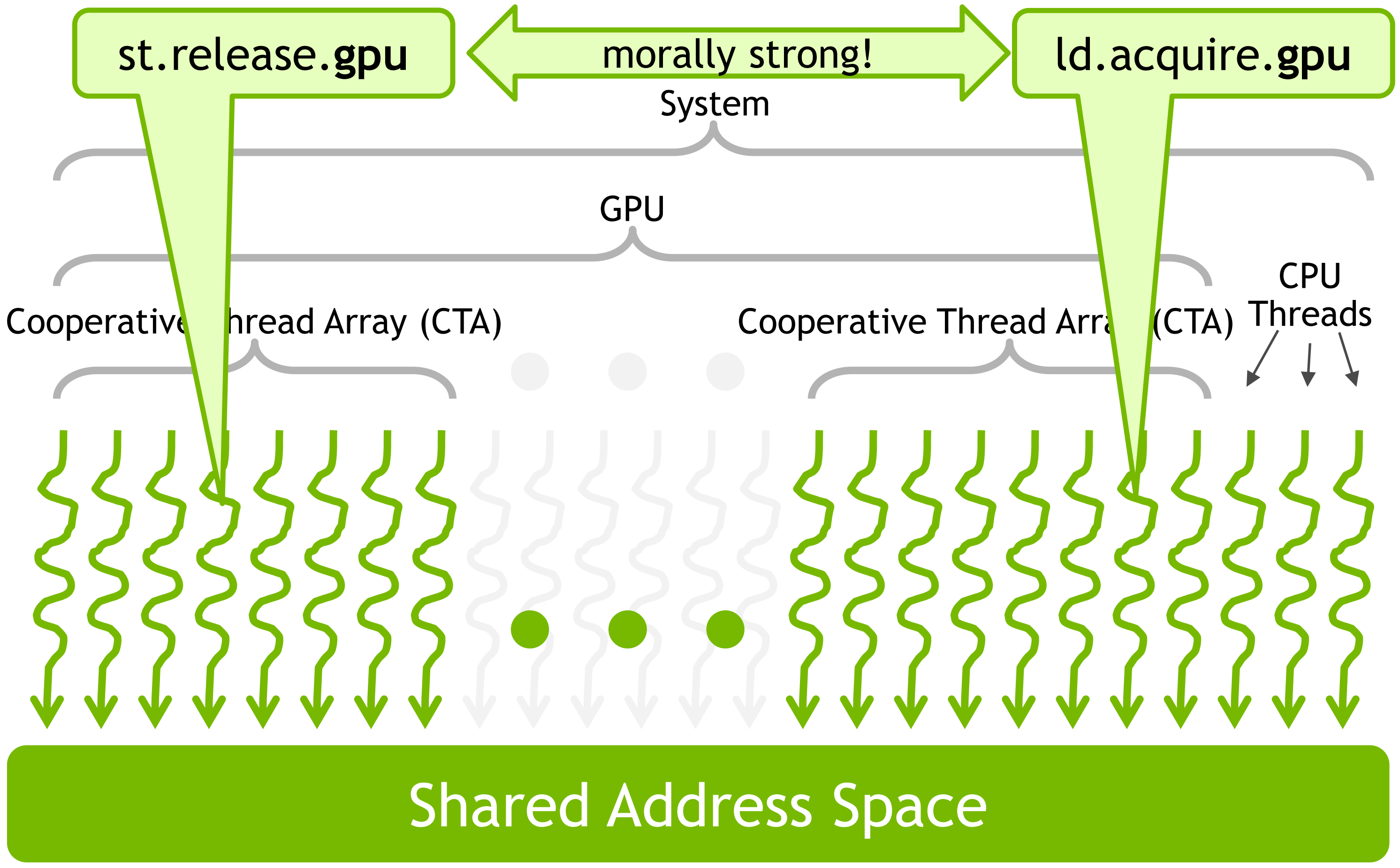


SCOPES AND MORAL STRENGTH


System
GPU
Cooperative Thread Array (CTA)
Cooperative Thread Array (CTA)
CPU Threads
Shared Address Space

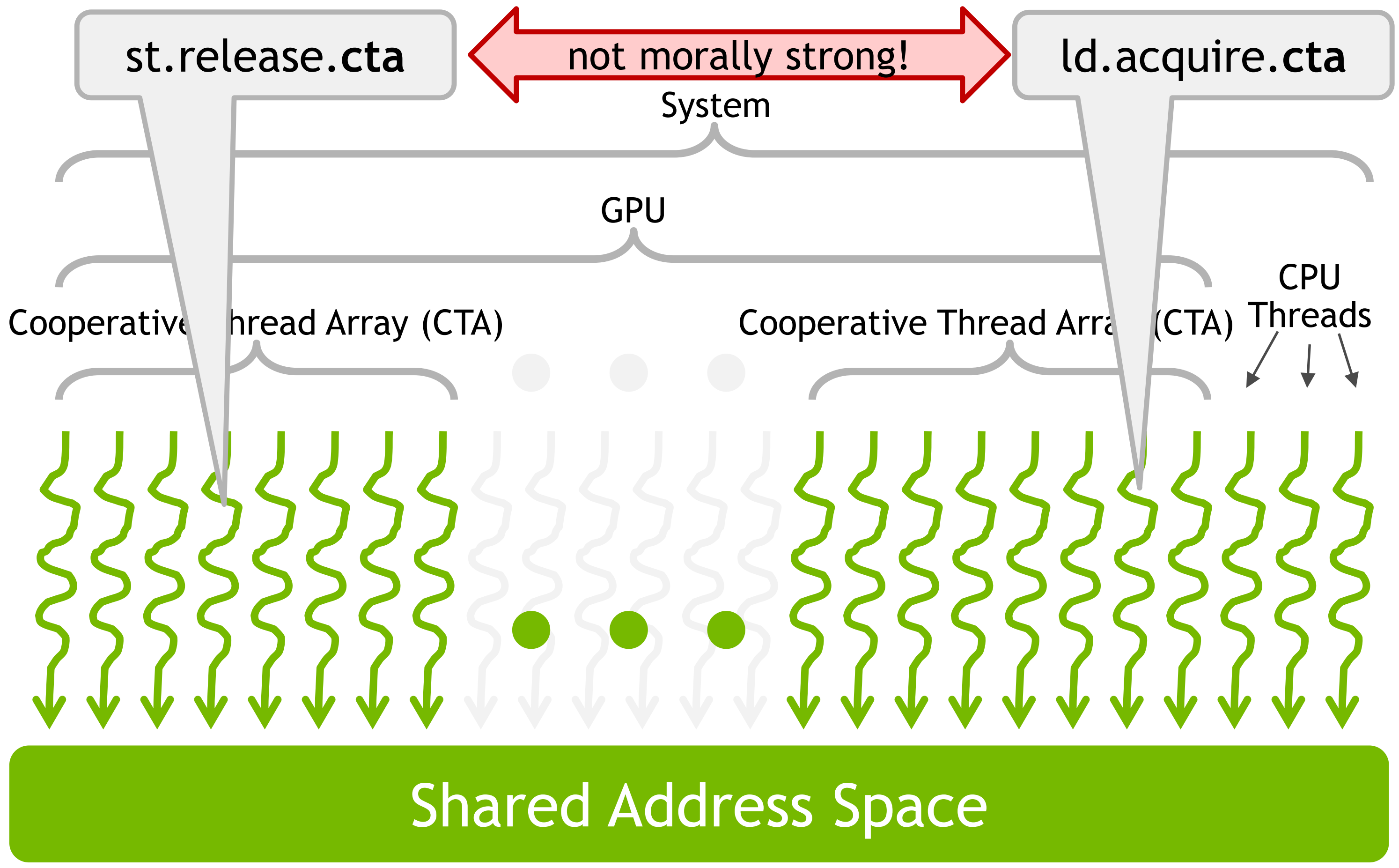
- A <u>scope</u> is a set of threads
- Each synchronizing memory instruction specifies a scope, e.g., “`ld.acquire.gpu`”
- <u>Morally strong</u> aka mutually scope inclusive: a pair of instructions where the scope of each includes the other

SCOPES AND MORAL STRENGTH



- A scope is a set of threads
- Each synchronizing memory instruction specifies a scope, e.g., “`ld.acquire.gpu`”
- Morally strong aka mutually scope inclusive: a pair of instructions where the scope of each includes the other

SCOPES AND MORAL STRENGTH



- A <u>scope</u> is a set of threads
- Each synchronizing memory instruction specifies a scope, e.g., "`ld.acquire.gpu`"
- <u>Morally strong</u> aka mutually scope inclusive: a pair of instructions where the scope of each includes the other

PTX MEMORY CONSISTENCY MODEL AXIOMS

The “Least Common Denominator”, plus scopes

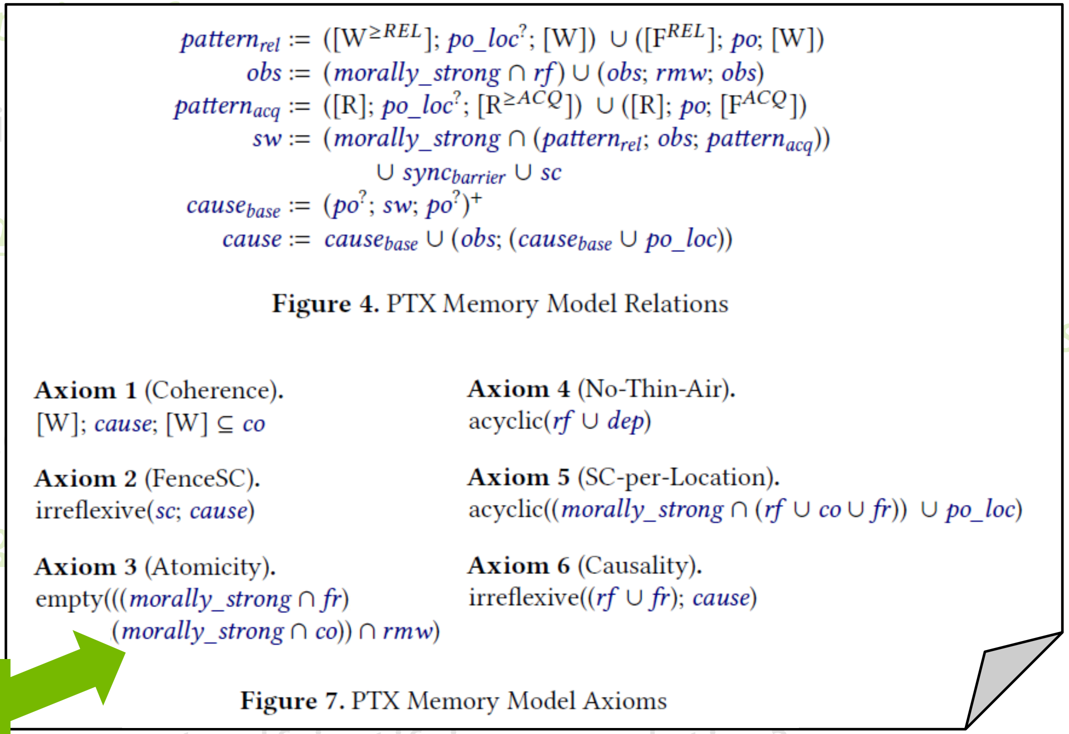
- **Causality**, *built out of morally strong pairs of memory accesses*
  - Release/Acquire, CTA Execution Barriers, Fences (transitively)
- **Coherence Order**, *for causally related/morally strong pairs of memory accesses*
- **Sequential Consistency per Location**, *for morally strong pairs of memory accesses*
  - expected single-threaded and same-address behavior
- **Atomicity of RMWs**, *for morally strong pairs of memory accesses*
- **No Out-of-Thin-Air Executions**
  - still-open theoretical problem: how to prevent self-justifying speculation?

Lustig, Sahasrabuddhe, Giroux, “A Formal Analysis of the NVIDIA PTX Memory Consistency Model”, ASPLOS 2019

PTX MEMORY CONSISTENCY MODEL AXIOMS

The “Least Common Denominator”, plus scopes

- **Causality**, *built out of morally strong p*
  - Release/Acquire, CTA Execution Barri
- **Coherence Order**, *for causally relatec*
- **Sequential Consistency per Location**, s
  - expected single-threaded and same-a
- **Atomicity of RMWs**, *for morally strong*
- **No Out-of-Thin-Air Executions**
  - still-open theoretical problem: how to prevent self-justifying speculation?



$$\begin{aligned}
\mathit{pattern}_{rel} &:= ([W^{\geq REL}]; \mathit{po\_loc}^?; [W]) \cup ([F^{REL}]; \mathit{po}; [W]) \\
\mathit{obs} &:= (\mathit{morally\_strong} \cap \mathit{rf}) \cup (\mathit{obs}; \mathit{rmw}; \mathit{obs}) \\
\mathit{pattern}_{acq} &:= ([R]; \mathit{po\_loc}^?; [R^{\geq ACQ}]) \cup ([R]; \mathit{po}; [F^{ACQ}]) \\
\mathit{sw} &:= (\mathit{morally\_strong} \cap (\mathit{pattern}_{rel}; \mathit{obs}; \mathit{pattern}_{acq})) \\
&\qquad \cup \mathit{sync}_{barrier} \cup \mathit{sc} \\
\mathit{cause}_{base} &:= (\mathit{po}^?; \mathit{sw}; \mathit{po}^?)^+ \\
\mathit{cause} &:= \mathit{cause}_{base} \cup (\mathit{obs}; (\mathit{cause}_{base} \cup \mathit{po\_loc}))
\end{aligned}$$

Figure 4. PTX Memory Model Relations

Axiom 1 (Coherence).
$[W]; \mathit{cause}; [W] \subseteq \mathit{co}$

Axiom 2 (FenceSC).
$\text{irreflexive}(\mathit{sc}; \mathit{cause})$

Axiom 3 (Atomicity).
$\text{empty}(((\mathit{morally\_strong} \cap \mathit{fr})\ (\mathit{morally\_strong} \cap \mathit{co})) \cap \mathit{rmw})$

Axiom 4 (No-Thin-Air).
$\text{acyclic}(\mathit{rf} \cup \mathit{dep})$

Axiom 5 (SC-per-Location).
$\text{acyclic}((\mathit{morally\_strong} \cap (\mathit{rf} \cup \mathit{co} \cup \mathit{fr})) \cup \mathit{po\_loc})$

Axiom 6 (Causality).
$\text{irreflexive}((\mathit{rf} \cup \mathit{fr}); \mathit{cause})$

Figure 7. PTX Memory Model Axioms

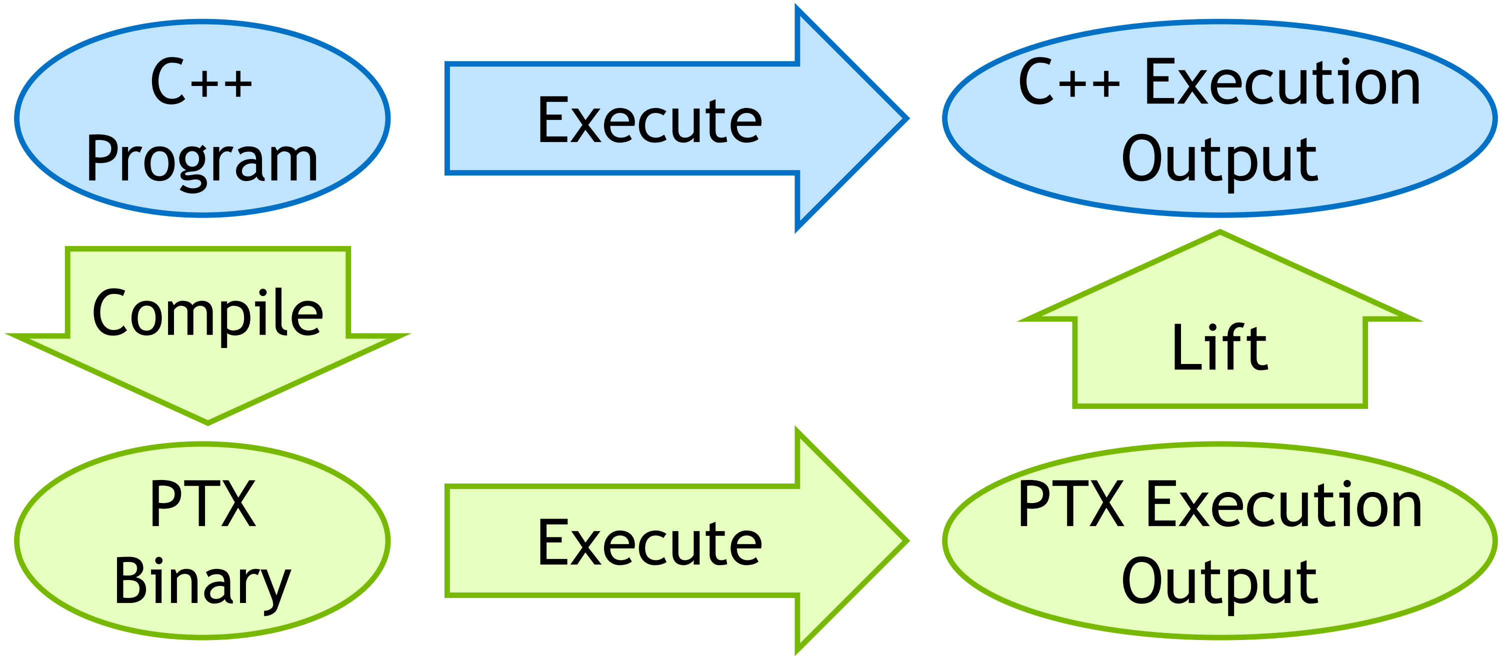
Full details in PTX documentation and in paper!

VERIFIED COMPILER MAPPINGS

How do we know the model is compatible with general-purpose languages like CUDA and C++?

Lustig, Sahasrabuddhe, Giroux, “A Formal Analysis of the NVIDIA PTX Memory Consistency Model”, ASPLOS 2019

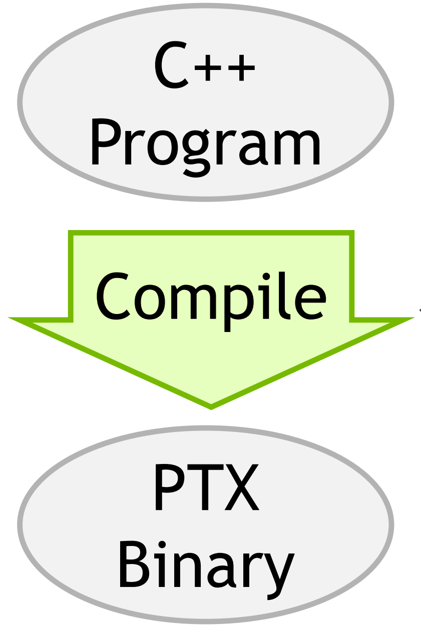
DEFINING AND VERIFYING "CORRECTNESS"



1. Execute according to abstract C++ semantics
2. Compile to PTX, execute, and interpret as execution of original C++ program

Correctness:
#2 must always be consistent with #1

C++ COMPILER MAPPINGS

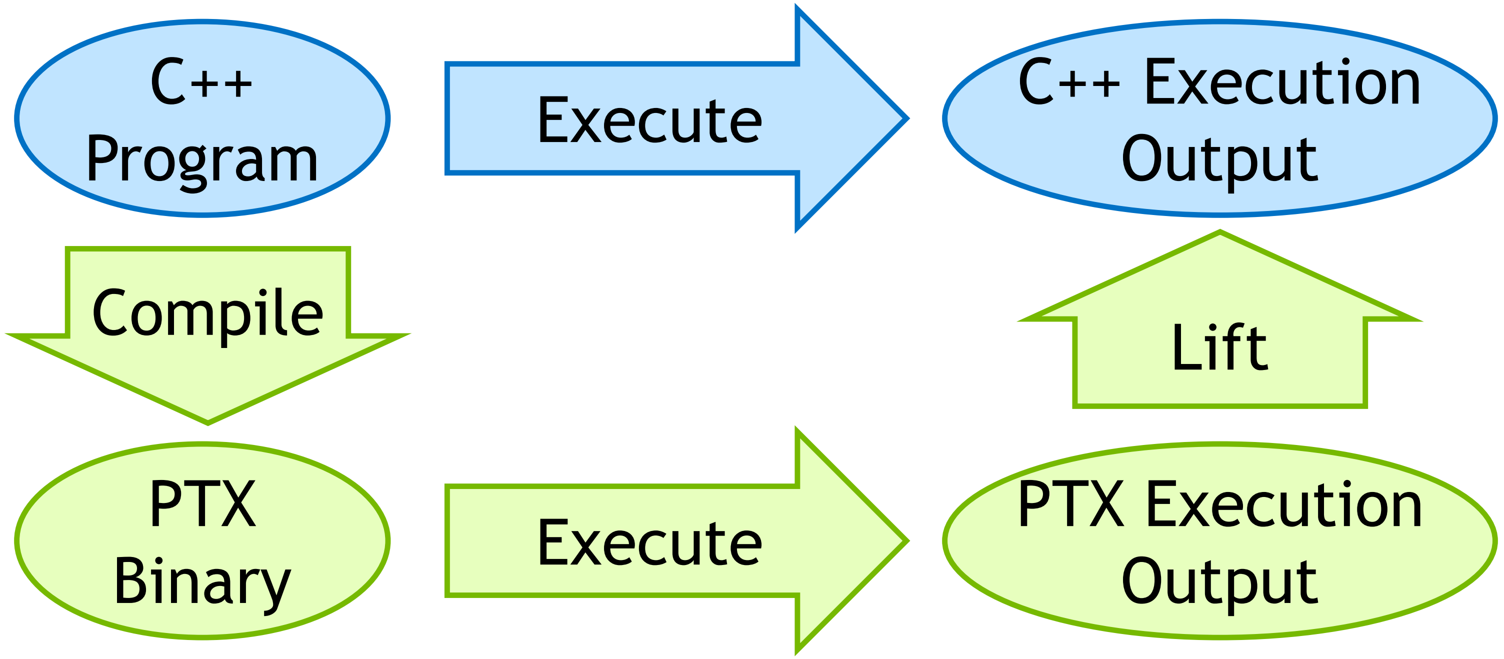


C++	PTX
non-atomic load	`ld.weak`
`atomic_load(`*addr*`, memory_order_relaxed,` *scope*`)`	`ld.relaxed.`*scope* *addr*
`atomic_load(`*addr*`, memory_order_acquire,` *scope*`)`	`ld.acquire.`*scope* *addr*
`atomic_load(`*addr*`, memory_order_seq_cst,` *scope*`)`	`fence.sc.`*scope*`; ld.acquire.`*scope* *addr*
...	...

Modern best practice:

Provide a fixed compiler mapping from C++ to PTX assembly

DEFINING AND VERIFYING “CORRECTNESS”

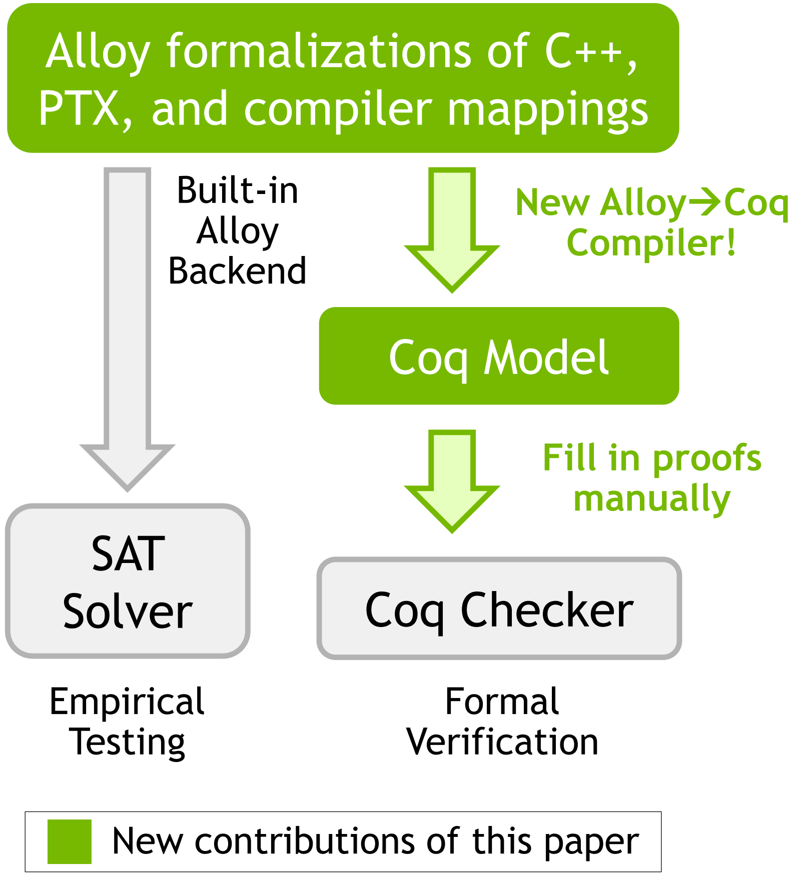


Build models of both paths, and then:

1. **Empirically test for counterexamples in the theory**
2. **Rigorously prove the theory correct**
3. Empirically test on real hardware
4. Whatever else!

VERIFYING OUR C++ COMPILER MAPPINGS

- We developed an **Alloy→Coq compiler**, enabling a unified framework combining:
  - Alloy: Relational Model Finder
  - Coq: Interactive Theorem Prover
- Empirically test AND formally prove that the compiler mappings are sound
  - Same model for both flows!



THE NVIDIA PTX MEMORY CONSISTENCY MODEL

Summary and Conclusion

- Scoped weak memory model, designed to maximize performance, architectural flexibility, and portability
- Allows GPU threads to freely communicate with any other thread in the system
    - Fixes bugs, livelock/starvation issues in past generations
- Developed a new unified framework to empirically test AND formally prove the correctness of C++ compiler mappings
    - Extends modern best practice for hardware memory consistency models

All materials available online: https://github.com/nvlabs/PTXMemoryModel