Provable GPU Data-Races in Static Race Detection

Characterizing **True Data-Race Alarms** in a **Behavioral Type**

Dennis Liew, Tiago Cogumbreiro, Julien Lange

PLACES’22, April 3rd 2022

Overview

Introduction

The CUDA Programming Model

CUDA is an extension of C, handling parallel code.

CUDA follows the Single-Instruction-Multiple-Threads (SIMT) model, all threads execute a copy of the GPU program (kernel).

*CUDA’s programming model is similar to OpenCL.

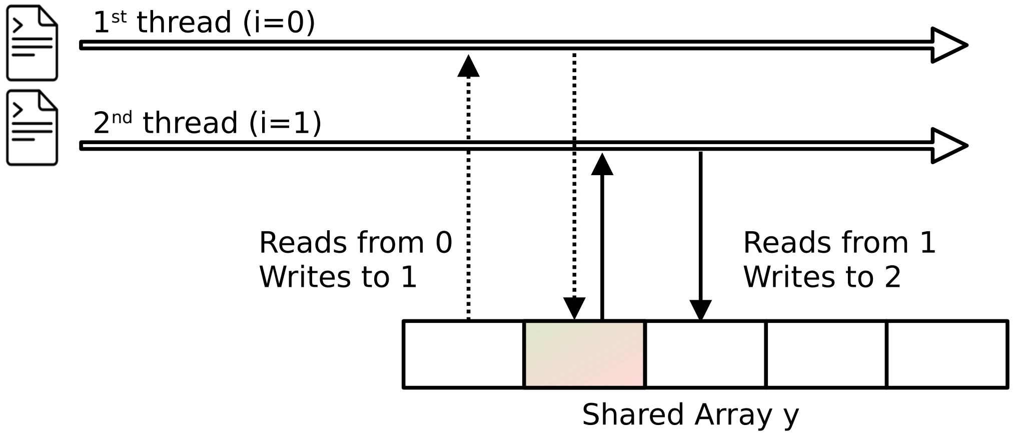
```
void saxpy(int n, float a, float *x, float *y)
{
  int i = blockIdx.x*blockDim.x + threadIdx.x;
  if (i < n) y[i] = a*x[i] + y[i];
}
```

Single-Precision A·X Plus Y (**saxpy**) is the “Hello World” of CUDA, our running example

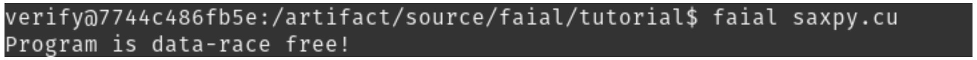
Data-races in CUDA

Data-races : When two or more threads access the same memory location, and at least one is a Write; causing unwanted nondeterminism.

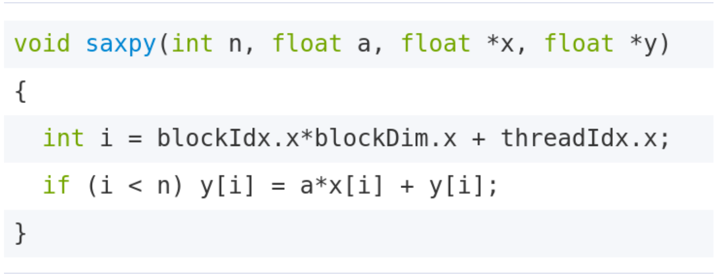
```
void saxpyracy(int n, float a, float *x, float *y)
{
  int i = blockIdx.x*blockDim.x + threadIdx.x;
  if (i < n) y[i+1] = a*x[i] + y[i];
}
```



Faial [CAV'21] : DRF Checker

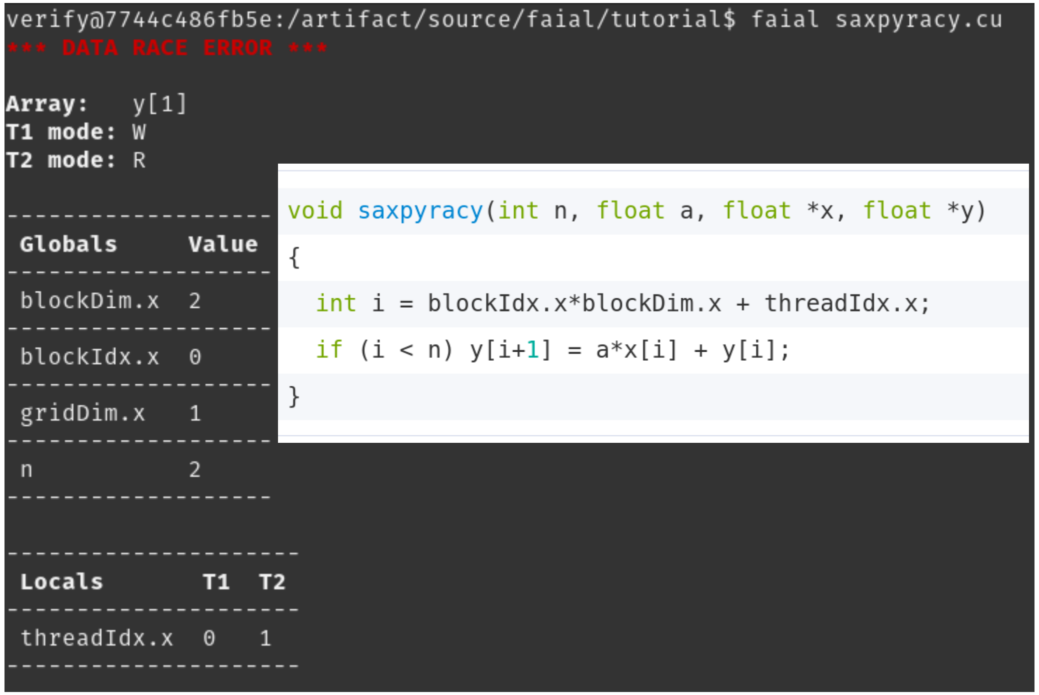


```
verify@7744c486fb5e:/artifact/source/faial/tutorial$ faial saxpy.cu
Program is data-race free!
```



```
void saxpy(int n, float a, float *x, float *y)
{
  int i = blockIdx.x*blockDim.x + threadIdx.x;
  if (i < n) y[i] = a*x[i] + y[i];
}
```

*Faial is **sound but incomplete.**

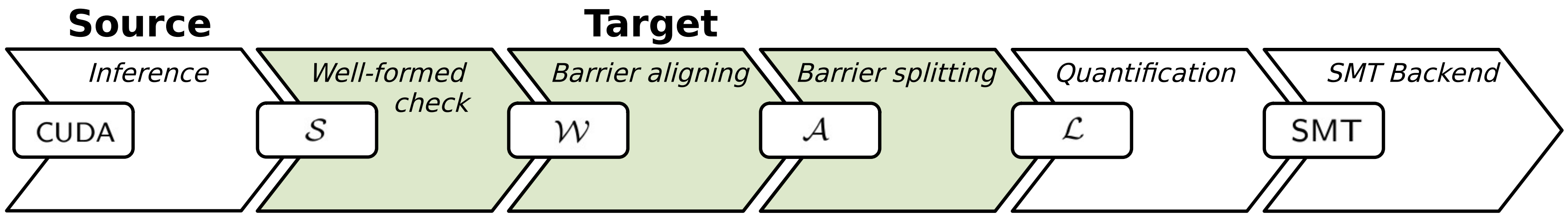


```
verify@7744c486fb5e:/artifact/source/faial/tutorial$ faial saxpyracy.cu
*** DATA RACE ERROR ***

Array:    y[1]
T1 mode:  W
T2 mode:  R

-------------------
 Globals      Value
-------------------
 blockDim.x   2
-------------------
 blockIdx.x   0
-------------------
 gridDim.x    1
-------------------
 n            2
-------------------


--------------------
 Locals       T1  T2
--------------------
 threadIdx.x  0   1
--------------------
```

```
void saxpyracy(int n, float a, float *x, float *y)
{
  int i = blockIdx.x*blockDim.x + threadIdx.x;
  if (i < n) y[i+1] = a*x[i] + y[i];
}
```

[CAV'21]: Tiago Cogumbreiro, Julien Lange, Dennis Liew & Hannah Zicarelli : Checking Data-Race Freedom of GPU Kernels, Compositionally.

Theory Behind Faial [CAV’21]



- a compositional analysis for DRF, based on memory access protocols (MAPs).
- protocols are **behavioral types** that codify the way threads interact over shared memory.
- mechanized proofs of our theoretical results.
- Faial outperforms the state-of- the-art, (verify at least 1.42× more real-world kernels).

False Alarms in Faial

Over approximates by ignoring array contents (what is being read from / written to arrays).

This over approximation in MAPs is what makes Faial scalable.

The downsides are **False Alarms** caused by **data-dependent** kernels.

```
void readindex (int* x)
{
    x[threadIdx.x] = threadIdx.x;
    int z = x[threadIdx.x];
    x[z] = 0;
}
```

CUDA

$\xrightarrow{\text{Over approximation}}$

```
wr[tid];
rd[tid];
wr[x]
```

Behavioral Type implemented in Faial

readindex is actually data-race free (DRF) but Faial reports it as Racy.

False Alarms in Static Analysis

Static Analyzers suffer from false positives (**false alarms**)

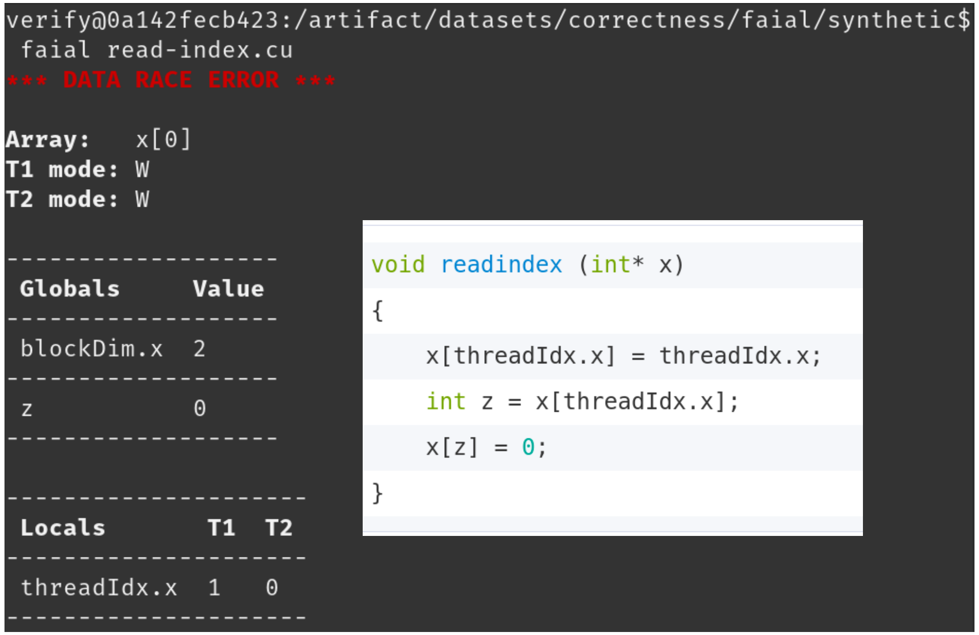
[CAV’21] evaluated several static analyzers on *readindex*, and most report a false alarm.

[ICSE’13], [CACM’18] show that false alarms hinder the adoption and use of static analyzers in industrial settings.

[ICSE’13]: Brittany Johnson, Yoonki Song, Emerson Murphy-Hill & Robert Bowdidge : Why Don’t Software Developers Use Static Analysis Tools to Find Bugs?

[CACM’18]: Caitlin Sadowski, Edward Aftandilian, Alex Eagle, Liam Miller-Cushon & Ciera Jaspan : Lessons from Building Static Analysis Tools at Google.

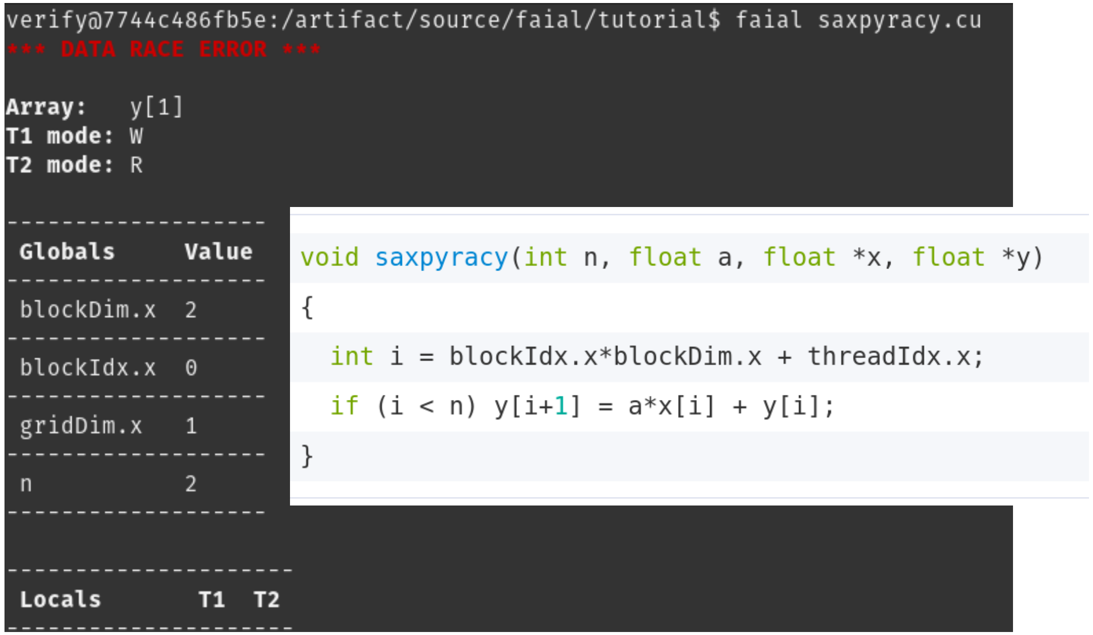
Can we prove certain alarms as True Alarms?



```
verify@0a142fecb423:/artifact/datasets/correctness/faial/synthetic$
 faial read-index.cu
*** DATA RACE ERROR ***

Array:    x[0]
T1 mode: W
T2 mode: W

-------------------
 Globals      Value
-------------------
 blockDim.x   2
-------------------
 z            0
-------------------

---------------------
 Locals        T1  T2
---------------------
 threadIdx.x   1   0
---------------------
```

```
void readindex (int* x)
{
    x[threadIdx.x] = threadIdx.x;
    int z = x[threadIdx.x];
    x[z] = 0;
}
```

Unknown



```
verify@7744c486fb5e:/artifact/source/faial/tutorial$ faial saxpyracy.cu
*** DATA RACE ERROR ***

Array:    y[1]
T1 mode: W
T2 mode: R

-------------------
 Globals      Value
-------------------
 blockDim.x   2
-------------------
 blockIdx.x   0
-------------------
 gridDim.x    1
-------------------
 n            2
-------------------

---------------------
 Locals        T1  T2
---------------------
```

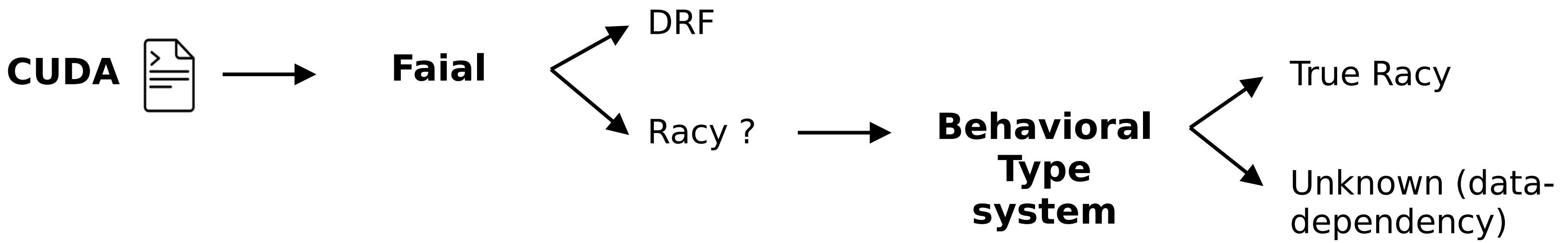
```
void saxpyracy(int n, float a, float *x, float *y)
{
  int i = blockIdx.x*blockDim.x + threadIdx.x;
  if (i < n) y[i+1] = a*x[i] + y[i];
}
```

Provably True Alarm

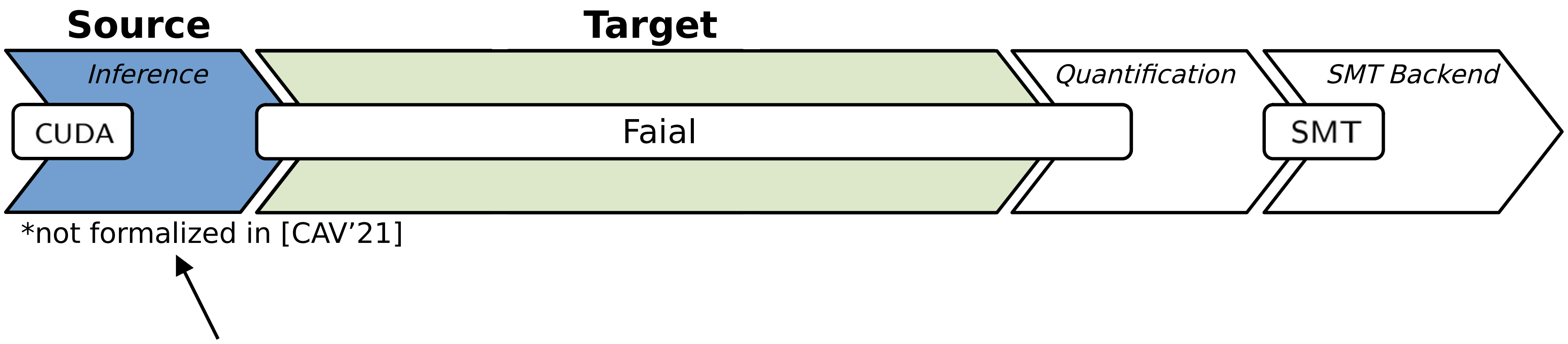
Our Approach

Solving the problem of **False Alarms**:

Specify a core language (BabyCUDA) and a behavioral type system, for which the analysis of Faial is sound and complete for well-typed programs.



Theoretical Pipeline



We are now formalizing it through BabyCUDA, representative subset of CUDA.

BabyCUDA: Syntax

Source Syntax: BabyCUDA

$$\begin{array}{rcl} \mathcal{B} \ni b & ::= & \mathtt{A}[n] := n \mid \mathsf{let}\ x = \mathtt{A}[n]\ \mathsf{in}\ b \\ & \mid & b\,;b \mid \mathsf{if}\ c\ \{b\}\ \mathsf{else}\ \{b\} \\ & \mid & \mathsf{for}\ x \in n..m\ \{b\} \mid \mathsf{skip} \end{array}$$

Simplified *saxpy* in CUDA

```
if (threadIdx.x < n) y[threadIdx.x] = y[threadIdx.x];
```

Simplified *saxpy* in BabyCUDA

```
if (tid < n) { let x = A[tid] in A[tid] := x }
else { skip }
```

Target Syntax: Memory Access Protocols (Faial)

$$\mathcal{U} \ni \quad u \quad ::= \quad \textsf{skip} \mid o[i] \mid u\,;u \mid \textsf{if}\ c\ \{u\}\ \textsf{else}\ \{u\} \mid \textsf{for}\ x \in n..m\ \{u\}$$

Following, we show how to infer a Memory Access Protocol from a BabyCUDA program

Simplified *saxpy* in BabyCUDA

```
if (tid < n) { let x = A[tid] in A[tid] := x }
else { skip }
```

Simplified *saxpy as a* Memory Access Protocol (Faial)

```
if (tid < n) { rd[tid]; wr[tid] }
else { skip }
```

BabyCUDA: Operational Semantics

BabyCUDA: Operational Semantics

$$\underbrace{A}_{\text{Input Accesses}}, \underbrace{b}_{\text{Program}} \downarrow_{\underbrace{H,i}_{\text{History \& Thread ID}}} \underbrace{A}_{\text{Output Accesses}}$$

BabyCUDA: Operational Semantics

Semantics

$$\boxed{lastwrite(j,H)=k} \quad \boxed{A,b \downarrow_{H,i} A} \quad \boxed{H,b\downarrow H}$$

LASTWRITE-CURR

$$\frac{\exists i\colon P(i)=(R,W) \qquad W(j)=k}{lastwrite(j,P::H)=k}$$

LASTWRITE-PREV

$$\frac{\forall i\colon P(i)=(R,W) \implies j\notin dom(W)}{lastwrite(j,P::H)=lastwrite(j,H)}$$

LASTWRITE-UNDEF

$$\frac{}{lastwrite(j,[\,])=\bot}$$

READ

$$\frac{n\downarrow j \qquad (R\cup\{j\},W),b[x:=lastwrite(j,\{i\colon (R,W)\}::H)]\downarrow_{H,i} B}{A, \mathsf{let}\ x = \mathtt{A}[n]\ \mathsf{in}\ b \downarrow_{H,i} B}$$

WRITE

$$\frac{n\downarrow j \qquad m\downarrow k}{(R,W),\mathtt{A}[n] := m \downarrow_{H,i} (R,W[j\mapsto k])}$$

SEQ

$$\frac{A,b_1 \downarrow_{H,i} B \quad B,b_2 \downarrow_{H,i} C}{A,b_1\,;b_2 \downarrow_{H,i} C}$$

IF-T

$$\frac{c\downarrow\mathtt{true} \quad A,b_1 \downarrow_{H,i} B}{A,\mathsf{if}\ c\ \{b_1\}\ \mathsf{else}\ \{b_2\} \downarrow_{H,i} B}$$

IF-F

$$\frac{c\downarrow\mathtt{false} \quad A,b_2 \downarrow_{H,i} B}{A,\mathsf{if}\ c\ \{b_1\}\ \mathsf{else}\ \{b_2\} \downarrow_{H,i} B}$$

FOR-1

$$\frac{(n\geq m)\downarrow\mathtt{true}}{A,\mathsf{for}^{\mathtt{U}}\ x\in n..m\ \{b\} \downarrow_{H,i} A}$$

FOR-2

$$\frac{(n<m)\downarrow\mathtt{true} \qquad A,b[x:=n] \downarrow_{H,i} B \qquad B,\mathsf{for}^{\mathtt{U}}\ x\in n+1..m\ \{b\} \downarrow_{H,i} C}{A,\mathsf{for}^{\mathtt{U}}\ x\in n..m\ \{b\} \downarrow_{H,i} C}$$

SKIP

$$\frac{}{A,\mathsf{skip} \downarrow_{H,i} A}$$

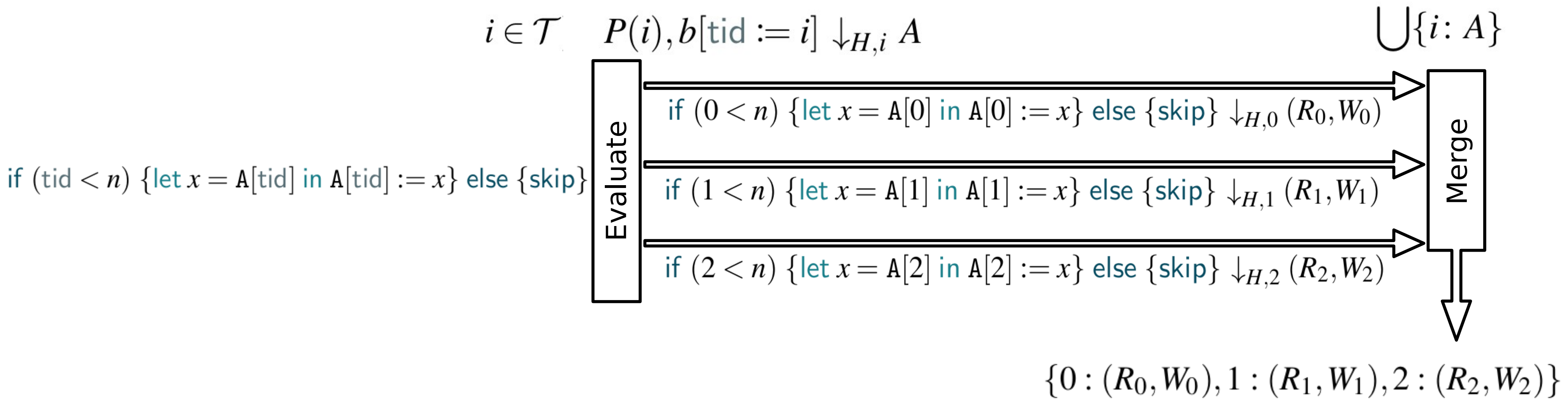
PAR

$$\frac{Q=\bigcup\{i\colon A \mid P(i), b[\mathsf{tid}:=i]\downarrow_{H,i} A \wedge i\in\mathcal{T}\}}{P::H,b\downarrow Q::H}$$

BabyCUDA: Parallel semantics

$$\frac{Q = \bigcup \{ i\colon A \mid P(i), b[\mathsf{tid} := i] \downarrow_{H,i} A \wedge i \in \mathcal{T} \}}{P::H, b \downarrow Q::H}\ \text{PAR}$$

- For every thread, evaluate the BabyCUDA program, *b yielding* accesses *A.*
- Replace tid with their unique thread identifier i
- Merge all *i:A* into the current phase.


$i \in \mathcal{T} \quad P(i), b[\mathsf{tid} := i] \downarrow_{H,i} A$
$\bigcup \{ i\colon A \}$
$\mathsf{if}\ (\mathsf{tid} < n)\ \{\mathsf{let}\ x = \mathtt{A}[\mathsf{tid}]\ \mathsf{in}\ \mathtt{A}[\mathsf{tid}] := x\}\ \mathsf{else}\ \{\mathsf{skip}\}$
Evaluate
$\mathsf{if}\ (0 < n)\ \{\mathsf{let}\ x = \mathtt{A}[0]\ \mathsf{in}\ \mathtt{A}[0] := x\}\ \mathsf{else}\ \{\mathsf{skip}\} \downarrow_{H,0} (R_0, W_0)$
$\mathsf{if}\ (1 < n)\ \{\mathsf{let}\ x = \mathtt{A}[1]\ \mathsf{in}\ \mathtt{A}[1] := x\}\ \mathsf{else}\ \{\mathsf{skip}\} \downarrow_{H,1} (R_1, W_1)$
$\mathsf{if}\ (2 < n)\ \{\mathsf{let}\ x = \mathtt{A}[2]\ \mathsf{in}\ \mathtt{A}[2] := x\}\ \mathsf{else}\ \{\mathsf{skip}\} \downarrow_{H,2} (R_2, W_2)$
Merge
$\{0 : (R_0, W_0), 1 : (R_1, W_1), 2 : (R_2, W_2)\}$

BabyCUDA: Type System

saxpy and *saxpyracy* are Well-typed

$$\underbrace{\mathcal{V}}_{\text{Variable Set}} \vdash \underbrace{b}_{\text{Program}} \blacktriangleright \underbrace{u}_{\text{Behavioral type}}$$

$$\underbrace{\{\mathsf{tid}, n\}}_{\text{Variable Set}} \vdash \underbrace{\mathsf{if}\ (\mathsf{tid} < n)\ \{\mathsf{let}\ x = \mathtt{A}[\mathsf{tid}]\ \mathsf{in}\ \mathtt{A}[\mathsf{tid}] := x\}\ \mathsf{else}\ \{\mathsf{skip}\}}_{\text{Program (BabyCUDA)}} \blacktriangleright \underbrace{\mathsf{if}\ (\mathsf{tid} < n)\ \{\mathsf{rd}[\mathsf{tid}]\ ;\mathsf{wr}[\mathsf{tid}]\}\ \mathsf{else}\ \{\mathsf{skip}\}}_{\text{Behavioral type (MAPs)}}$$

Well-typed + Racy = True Race

$$\underbrace{\{\mathsf{tid}, n\}}_{\text{Variable Set}} \vdash \underbrace{\mathsf{if}\ (\mathsf{tid} < n)\ \{\mathsf{let}\ x = \mathtt{A}[\mathsf{tid}]\ \mathsf{in}\ \mathtt{A}[\mathsf{tid}+1] := x\}\ \mathsf{else}\ \{\mathsf{skip}\}}_{\text{Program (BabyCUDA)}} \blacktriangleright \underbrace{\mathsf{if}\ (\mathsf{tid} < n)\ \{\mathsf{rd}[\mathsf{tid}]\ ;\mathsf{wr}[\mathsf{tid}+1]\}\ \mathsf{else}\ \{\mathsf{skip}\}}_{\text{Behavioral type (MAPs)}}$$

BabyCUDA: Behavioral type system

$\boxed{\mathcal{V} \vdash n}$ $\boxed{\mathcal{V} \vdash c}$ $\boxed{\mathcal{V} \vdash b \blacktriangleright u}$

T-N
$$\frac{fv(n) \subseteq \mathcal{V}}{\mathcal{V} \vdash n}$$

T-B
$$\frac{fv(c) \subseteq \mathcal{V}}{\mathcal{V} \vdash c}$$

T-WRITE
$$\frac{\mathcal{V} \vdash n}{\mathcal{V} \vdash \mathtt{A}[n] := m \blacktriangleright \mathsf{wr}[n]}$$

T-READ
$$\frac{\mathcal{V} \vdash n \qquad y \notin \mathcal{V} \qquad \mathcal{V} \vdash b \blacktriangleright u}{\mathcal{V} \vdash \mathsf{let}\ y = \mathtt{A}[n]\ \mathsf{in}\ b \blacktriangleright \mathsf{rd}[n]\ ;\ u}$$

T-SEQ
$$\frac{\mathcal{V} \vdash b_1 \blacktriangleright u_1 \qquad \mathcal{V} \vdash b_2 \blacktriangleright u_2}{\mathcal{V} \vdash b_1\ ; b_2 \blacktriangleright u_1\ ; u_2}$$

T-IF
$$\frac{\mathcal{V} \vdash c \qquad \mathcal{V} \vdash b_1 \blacktriangleright u_1 \qquad \mathcal{V} \vdash b_2 \blacktriangleright u_2}{\mathcal{V} \vdash \mathsf{if}\ c\ \{b_1\}\ \mathsf{else}\ \{b_2\} \blacktriangleright \mathsf{if}\ c\ \{u_1\}\ \mathsf{else}\ \{u_2\}}$$

T-FOR
$$\frac{\mathcal{V} \vdash n \qquad \mathcal{V} \vdash m \qquad x \notin \mathcal{V} \qquad \mathcal{V} \cup \{x\} \vdash b \blacktriangleright u}{\mathcal{V} \vdash \mathsf{for}^{\mathsf{U}}\ x \in n..m\ \{b\} \blacktriangleright \mathsf{for}^{\mathsf{U}}\ x \in n..m\ \{u\}}$$

T-SKIP
$$\frac{}{\mathcal{V} \vdash \mathsf{skip} \blacktriangleright \mathsf{skip}}$$

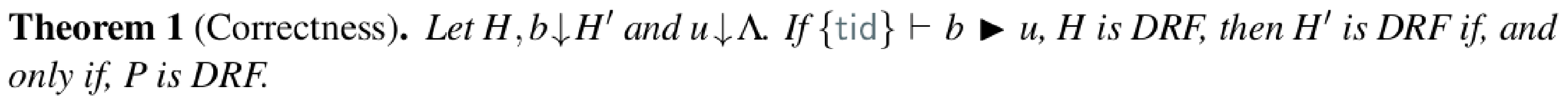
saxpyracy well-typed derivation

$$\dfrac{\dfrac{\dfrac{}{\{\mathsf{tid}, n\} \vdash \mathtt{A}[\mathsf{tid}+1] := x \blacktriangleright \mathsf{wr}[\mathsf{tid}+1]}\ \textsc{Write}}{\{\mathsf{tid}, n\} \vdash \mathsf{let}\ x = \mathtt{A}[\mathsf{tid}]\ \mathsf{in}\ \mathtt{A}[\mathsf{tid}+1] := x \blacktriangleright \mathsf{rd}[\mathsf{tid}]\ ;\ \mathsf{wr}[\mathsf{tid}+1]}\ \textsc{Read} \qquad \dfrac{}{\{\mathsf{tid}, n\} \vdash \mathsf{skip} \blacktriangleright \mathsf{skip}}\ \textsc{Skip}}{\{\mathsf{tid}, n\} \vdash \mathsf{if}\ (\mathsf{tid} < n)\ \{\mathsf{let}\ x = \mathtt{A}[\mathsf{tid}]\ \mathsf{in}\ \mathtt{A}[\mathsf{tid}+1] := x\}\ \mathsf{else}\ \{\mathsf{skip}\} \blacktriangleright \mathsf{if}\ (\mathsf{tid} < n)\ \{\mathsf{rd}[\mathsf{tid}]\ ;\ \mathsf{wr}[\mathsf{tid}+1]\}\ \mathsf{else}\ \{\mathsf{skip}\}}\ \textsc{If}$$

readindex ill-typed derivation

$$\dfrac{\dfrac{}{\{\mathsf{tid}\} \vdash \mathtt{A}[\mathsf{tid}] := \mathsf{tid} \blacktriangleright \mathsf{wr}[\mathsf{tid}]}\,\text{WRITE} \qquad \dfrac{\dfrac{}{\{\mathsf{tid}\} \nvdash \mathtt{A}[x] := 0 \blacktriangleright \mathsf{wr}[x]}\,\text{WRITE}}{\{\mathsf{tid}\} \nvdash \mathsf{let}\ x = \mathtt{A}[\mathsf{tid}]\ \mathsf{in}\ \mathtt{A}[x] := 0 \blacktriangleright \mathsf{rd}[\mathsf{tid}]\ ;\ \mathsf{wr}[x]}\,\text{READ}}{\{\mathsf{tid}\} \nvdash \mathtt{A}[\mathsf{tid}] := \mathsf{tid}\ ;\ \mathsf{let}\ x = \mathtt{A}[\mathsf{tid}]\ \mathsf{in}\ \mathtt{A}[x] := 0 \blacktriangleright \mathsf{wr}[\mathsf{tid}]\ ;\ \mathsf{rd}[\mathsf{tid}]\ ;\ \mathsf{wr}[x]}\,\text{SEQ}$$

BabyCUDA: Main result



Theorem 1 (Correctness). *Let $H, b \downarrow H'$ and $u \downarrow \Lambda$. If $\{\mathsf{tid}\} \vdash b \blacktriangleright u$, H is DRF, then H' is DRF if, and only if, P is DRF.*

Well-typed BabyCUDA programs are analyzed correctly (soundly and completely) by Faial.

Memory Access Protocols of well-typed programs preserve and reflect the concurrent accesses of the source program.

Conclusion

Related Work

[POPL’19] introduce the first DRF static analysis for **multithreaded** programs that is sound and complete, for a subset of all programs.

*generally inapplicable (or irrelevant) to GPU programming

[POPL’15] develop a deductive system to prove completeness of program analyses over an abstract domain.

[POPL’19]: Nikos Gorogiannis, Peter W. O’Hearn & Ilya Sergey (2019): A True Positives Theorem for a Static Race Detector.

[POPL’15]: Roberto Giacobazzi, Francesco Logozzo & Francesco Ranzato (2015): Analyzing Program Analyses.

Conclusion and Future Work

Introduced a **behavioral type system** that characterizes **true data-races** in Memory Access Protocols (Faial).

Main result guarantees that Faial’s analysis are sound and complete.

Future work include:
completing the Coq formalization and implementation.

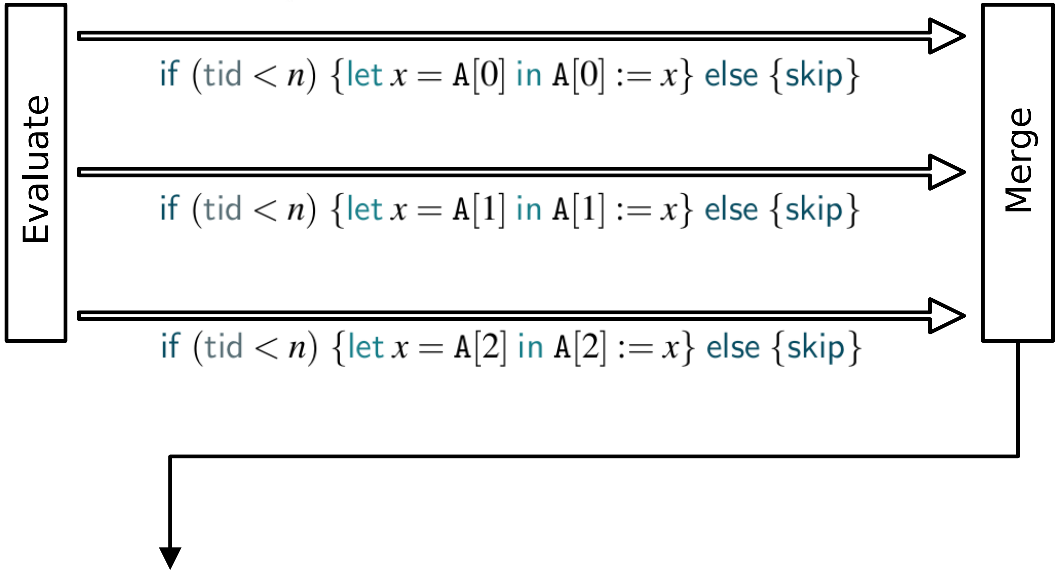
having an empirical evaluation of the type system’s applicability.

BabyCUDA

END

$$i \in \mathcal{T} \qquad P(i), b[\text{tid} := i] \downarrow_{H,i} A \qquad \bigcup\{i\colon A\}$$

if $(\text{tid} < n)$ $\{$let $x = \texttt{A}[\text{tid}]$ in $\texttt{A}[\text{tid}] := x\}$ else $\{$skip$\}$



if $(\text{tid} < n)$ $\{$let $x = \texttt{A}[0]$ in $\texttt{A}[0] := x\}$ else $\{$skip$\}$

if $(\text{tid} < n)$ $\{$let $x = \texttt{A}[1]$ in $\texttt{A}[1] := x\}$ else $\{$skip$\}$

if $(\text{tid} < n)$ $\{$let $x = \texttt{A}[2]$ in $\texttt{A}[2] := x\}$ else $\{$skip$\}$

$$[\{0\colon (\{0\}, \{0:0\}), 1\colon (\{1\}, \{1:1\}), 2\colon (\{2\}, \{2:2\})\}]$$

```
void readindex (int* x)
{
    x[threadIdx.x] = threadIdx.x;
    int z = x[threadIdx.x];
    x[z] = 0;
}
```

$$\{\mathsf{tid}\} \nvdash \mathtt{A}[\mathsf{tid}] := \mathsf{tid}\ ;\ \mathsf{let}\ x = \mathtt{A}[\mathsf{tid}]\ \mathsf{in}\ \mathtt{A}[x] := 0 \blacktriangleright \mathsf{wr}[\mathsf{tid}]\ ;\ \mathsf{rd}[\mathsf{tid}]\ ;\ \mathsf{wr}[x]$$