3. Operational semantics

The authors aim to provide the operational semantics to check the correctness of the warp specialized programs in terms of them being deadlock free, free from data races, being able to properly use the named barriers as they are limited in number hardware resources and being able to use the shared memory in the synchronized manner within individual CTA or thread blocks. They clearly state that they do not provide the same for the synchronized operation of warps in different CTAs. They assume that single CTA executes single program and that they have to provide the operational semantics for checking correctness within the single CTA. They also rely upon the fact that there is no interference of the global memory in the synchronization of the warps within the CTA.

3.1 Syntax

A CTA can contain typically 32 to 1024 threads that must synchronize to use the shared memory without leaving it in an inconsistent state. They denote the thread program by P and use P1 || P2 || …|| PN denote a CTA with threads P1 || P2 || …|| PN . They denote variables in shared memory as g and that they occupy only 64 bits. They abstract away all the sections of code except the synchronization and shared memory access commands. Each thread has thread id as *id* and *i* ranges over all the identifiers, B is used to denote named barriers and b ranges over all the named barriers. Each thread program has the grammar

P ::= return | c; P

c ::= read g| write g| arrive b n| sync b n

A thread program finishes its execution at a return command. Commands of the thread program can be of form read g to read, write to write a shard variable. Arrive and sync are synchronization operations called on barrier b and n is the expected number of threads to register at that named barrier. They assume that all the read and write commands are no-ops and are only helpful in detecting the data races.

The thread barrier sync 0 N is the sync across all the threads in a CTA. They also do not assume warp synchronous execution which is an assumption that all the warps execute in lock step (in adherence to the other threads). The sync command can be inserted after each command to simulate the lock step execution of all the thread.

3.2 State

The state of the CTA consists of two components:

1. An enabled map E that maps thread identifiers to booleans

signifying whether the thread is enabled or not. Threads are

disabled when they block on a barrier.

2. A barrier map B that maps barrier names to a triple consisting

of a list I of threads that have synced at the barrier, a list A of

threads that have arrived at the barrier, and the thread count,

describing the number of threads the barrier is expecting to

register if it is configured.

* An unconfigured barrier is B([], [], ⟂)
* A[x/y] is a map such that all the inputs agree with A except y that map to x.
* A[x/Y] is a map such that all the inputs agree with A except y belongs to Y map to x.
* ite(e1, e2, e3) denotes that “if e1 then e2 else e3”.

Initially, all the barriers are unconfigured state i.e for all b B([], [], ⟂) and all the threads are enabled i.e for all i E(i) = true.

Commands and their meanings:





