Add to website:

• I would have liked to see a table summarizing, which properties the other tools support.

\textbf{Existing GPU Verifiers}. GPUVerify~\cite{betts:2012} is based on Synchronous Delayed Visibility, which focus on detecting data race and barrier divergence, while reducing kernel verification procedures to the analysis of sequential programs. Symbolic Executor with Static Analysis (SESA)~\cite{sesa:2014} and GPU + KLEE (GKLEE)~\cite{Li:2012} are concrete plus symbolic (\textit{concolic}) execution tools. Initially, GKLEE modeled every original thread, which was later improved by taking into account symmetry, and it considers both kernel and main function, while checking barrier synchronization and data race, among others. SESA combines equivalent flows, which results in great performance improvement, focus on data race detection, and does not verify the main function. Prover of User GPU Programs (PUG)~\cite{Li:2010} is based on SMT solvers and applies symbolic static analysis to each kernel, but requires user annotations. CIVL~\cite{civl:2015} generates an abstract syntax tree and supports analysis and transformations. Besides, it also verifies concurrent programs with partial order reduction, which makes it very similar to the solution proposed here. Given that its support to CUDA libraries is still under development, CIVL is an interesting approach for checking simple CUDA programs.

%The chosen benchmark suite comprises $20$ CUDA kernels from NVIDIA GPU Computing SDK v$2$.$0$, $20$ CUDA kernels from Microsoft C$++$ AMP Sample Projects%~\cite{microsoft:2012}

%, and $104$ CUDA-based programs that explore a wide range of CUDA functionalities~\cite{cuda:2012}. In summary, the chosen suite contains $47.4$\% bug-free and $52.6$\% buggy benchmarks, which tackle data race, arithmetic operations, pointer assignment, {\tt \\_\\_device\\_\\_} function calls, general ANSI-C functions ({\it e.g.}, \texttt{memset}), general CUDA functions ({\it e.g.}, \texttt{cudaMemcpy}), general libraries in CUDA ({\it e.g.}, \texttt{curand.h}), and the ability to work with variables, type modifiers ({\it e.g.}, \texttt{unsigned}), pointers, type definitions, and intrinsic CUDA variables ({\it e.g.}, \texttt{uint4}) \textcolor{red}{poderiamos colocar toda essa descrição dos benchmarks numa pagina online e cita-la aqui??}.