Towards a Formally Verified Skeleton-based 
Data Parallel DSL for GPGPU
Izumi Asakura, Hidehiko Masuhara, Tomoyuki Aotani
(Tokyo Institute of Technology)

**Background & Our Work**

<table>
<thead>
<tr>
<th>App. level Verification</th>
<th>Sequential Lang.</th>
<th>Low-level GPGPU Lang. (e.g., CUDA)</th>
<th>High-level Lang. for GPGPU (e.g., Accelerate)</th>
</tr>
</thead>
</table>

**Technical Contributions:**
- An Approach to Formally Verified Compiler for Coq-embedded DSLs (compilation by proving)
- Compiler: reification and verified compiler in pure Coq
- Compositional Verification of Template-based Codegen.

**Compilation by Proving**
- Proving a lemma: there exists a GPGPU program equivalent to the source function
- Applying the compiler correctness lemma: Lemma compileOK: \( s \models \text{compileIR} s \)

**Programming in CertSkel**
- Source program: A pure Coq function on lists written in skeletons
- Compilating by proving: Proving a lemma: there exists a GPGPU program equivalent to the source function
- Compiling the GPGPU program (CUDA C) by Coq extraction mechanism
- Extracting GPGPU program (CUDA C) by Coq extraction mechanism
- The extracted OCaml program saves program to "./argmax.cu"

**compileIR:** a Certified Template-based Code Generator (written in Coq)

**Compositional correctness conditions**

**What We Have Done**
- Implemented: all compiler components & simple fusion trans. by rewriting (done before reification)
- Proved: \( \approx \) for some skels. (map, reduce) and seq. func. compiler
- By using helper tactic library GPUVeLib (symbolic execution on statements)

**Future Work**
- Finish the proof (top level function compilation)
- Precise semantics: machine integer does not have infinite precision
- More optimizations (e.g., fusion transformation)
- More features (e.g., nested parallelism, advanced skeletons)