End-to-End Verification for Subgraph Solving

Stephan Gocht[1,2], Ciaran McCreesh[3], Magnus O. Myreen[4], Jakob Nordström[2,1], **Andy Oertel**[1,2], Yong Kiam Tan[5]

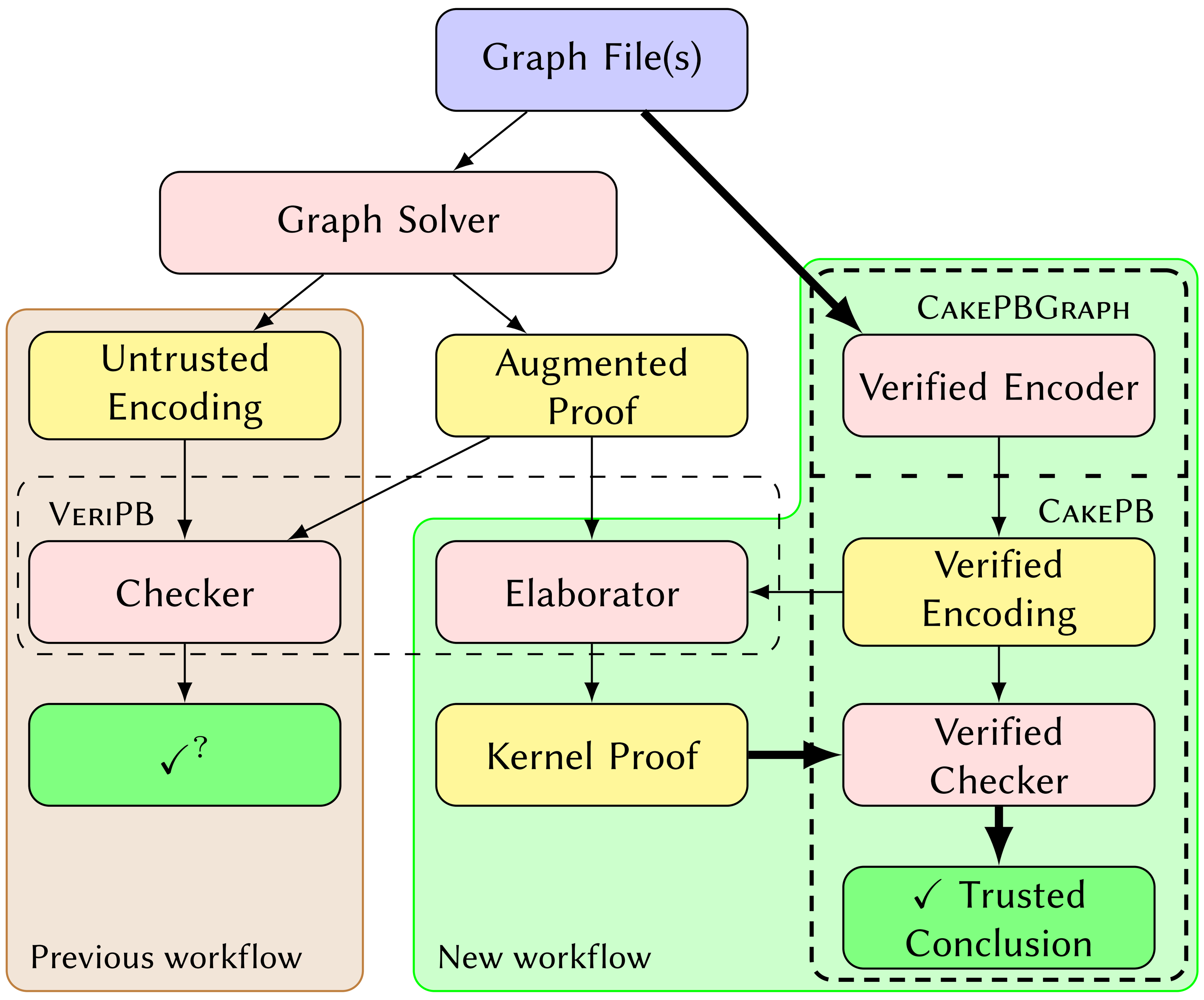
[1]Lund University [2]University of Copenhagen [3]University of Glasgow [4]Chalmers University [5]I^{2}R, A*STAR

Context: Modern subgraph solvers consist of thousands of lines of highly optimized code. How can we trust this code? Solver outputs proof that their result is correct.

Problem: Users have to trust the proof checker and the translation of the high-level graph problem into a 0-1 integer linear program (ILP) used for the proofs.

Solution: We close this issue by implementing a formally verified proof checker that can check a subset of the rules used in the proof system.

Our Workflow



Our workflow to get a formally verified result is as follows:

1. Solve problem with solver and generate augmented proof.
2. Elaborate augmented proof with VeriPB to kernel proof.
3. Check kernel proof with formally verified CakePB.

Trusted Base

Our workflow reduces the components that need to be trusted to:

- Higher-order logic (HOL) definitions of input parser and problems ⟶ easy to check
- HOL model of CakeML environment and correspondence to real system ⟶ validated extensively
- HOL4 theorem prover, including its logic, implementation, and execution environment ⟶ well established

Such a trusted base gives the highest assurance standard for formally verified software.

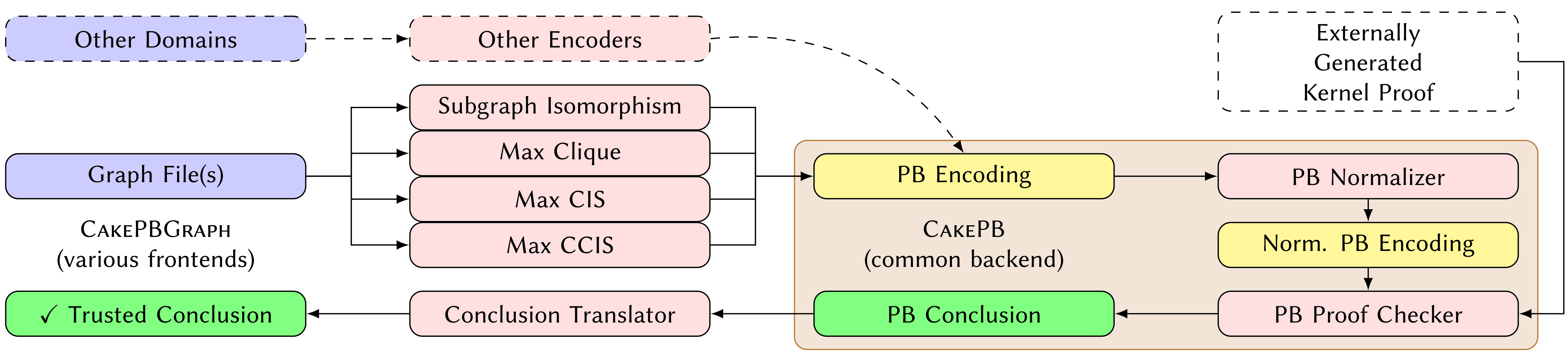
Proof Elaboration

The VeriPB proof format comes in two versions.

- Augmented proof format: Contains syntactic sugar for easy proof logging in the solver.
- Kernel proof format: Subset of augmented format that is efficient to check in a formally verified checker.

VeriPB can elaborate an augmented proof to a kernel proof.

Extensible Checking Framework



- Common backend: Performs general reasoning with 0-1 ILPs (a.k.a. PB).
- Frontend: Translates specific problem class into 0-1 ILP and back.

Experimental Results

- Our workflow is practicably viable for modern subgraph solvers.
- Elaborating augmented proof is not substantially slower than checking.
- Checking kernel proof about the same time as elaborating on average.

Source Code

Glasgow Subgraph Solver

VeriPB (elaborator)

CakePB (verified checker)