

END-TO-END VERIFICATION FOR SUBGRAPH SOLVING

Stephan Gocht^{1,2}, Ciaran McCreesh³, Magnus O. Myreen⁴,
Jakob Nordström^{2,1}, **Andy Oertel**^{1,2}, Yong Kiam Tan⁵

¹Lund University ²University of Copenhagen ³University of Glasgow ⁴Chalmers University ⁵I²R, A*STAR

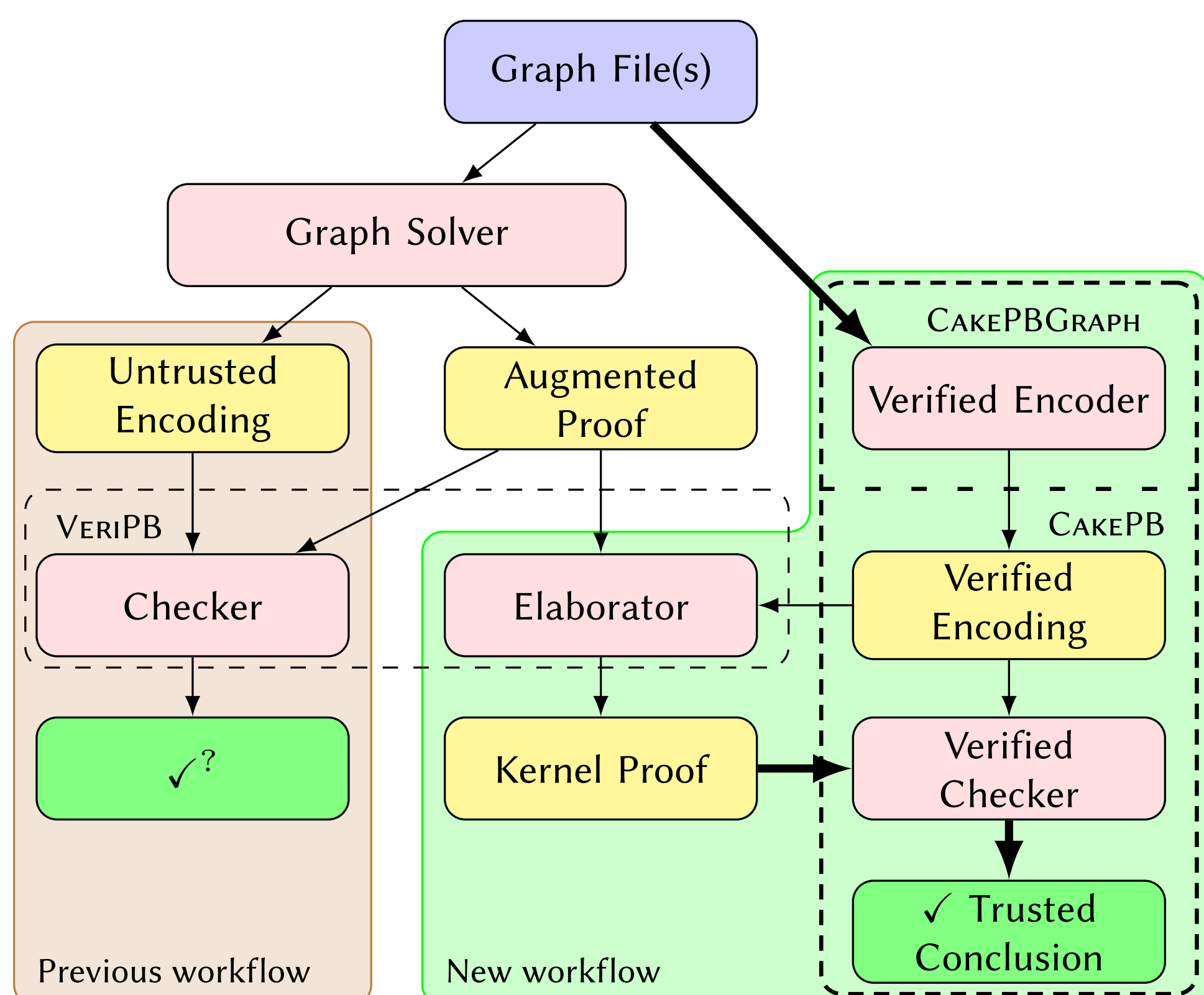


Context: Modern subgraph solvers consist of thousands of lines of highly optimized code. How can we trust this code? Solver outputs proof that their result is correct.

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.

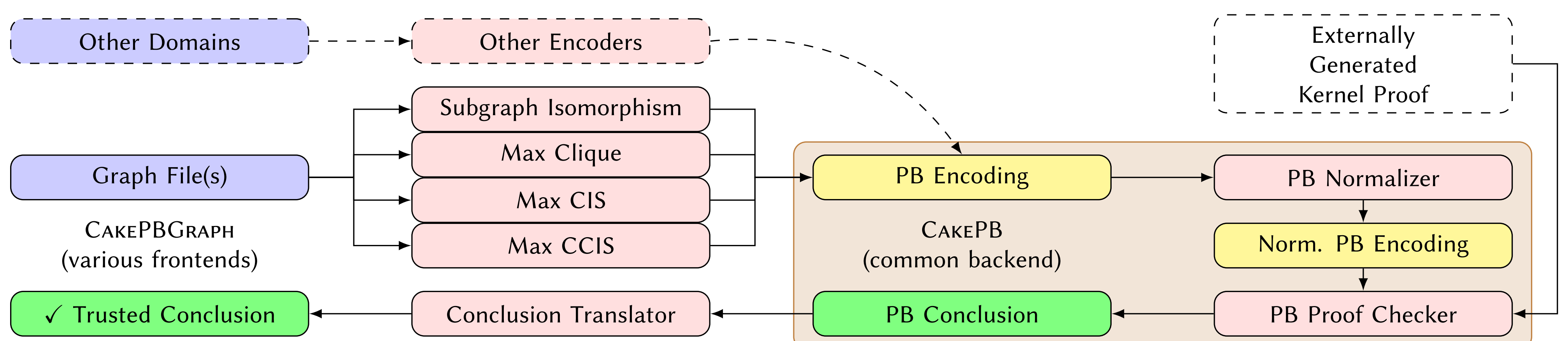
Proof Elaboration

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.

Source Code

Experimental Results

- Our workflow is **practically 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.



GLASGOW
SUBGRAPH SOLVER



VERIPB
(elaborator)



CAKEPB
(verified checker)