Ultimate TestGen: Cooperation in the Parallel Trace Abstraction Framework

Max Barth[1], Daniel Dietsch[2], Matthias Heizmann[3], and Marie-Christine Jakobs[1]

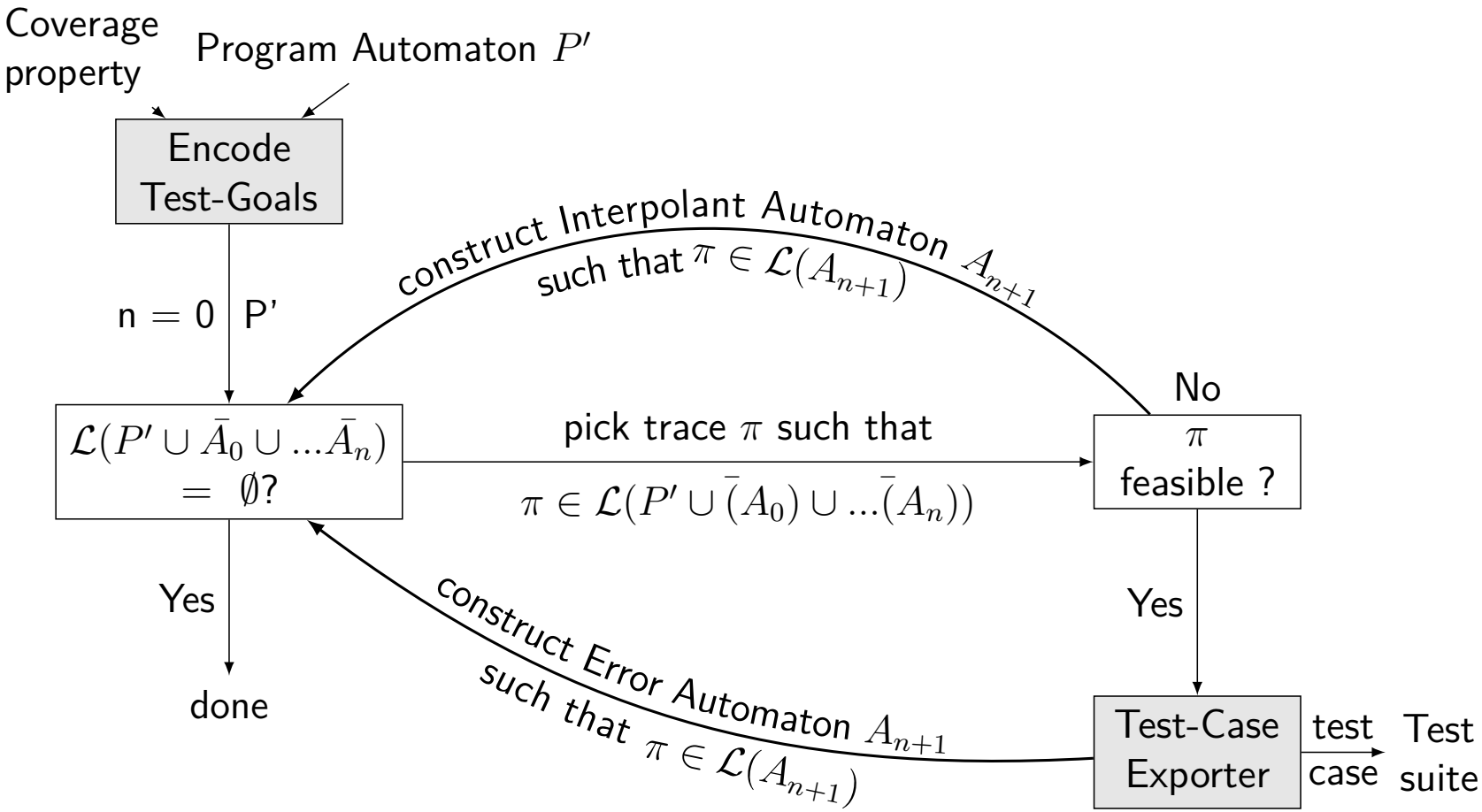
LMU Munich, Munich, Germany
University of Freiburg, Freiburg, Germany
University of Stuttgart, Stuttgart, Germany

2026-04-11

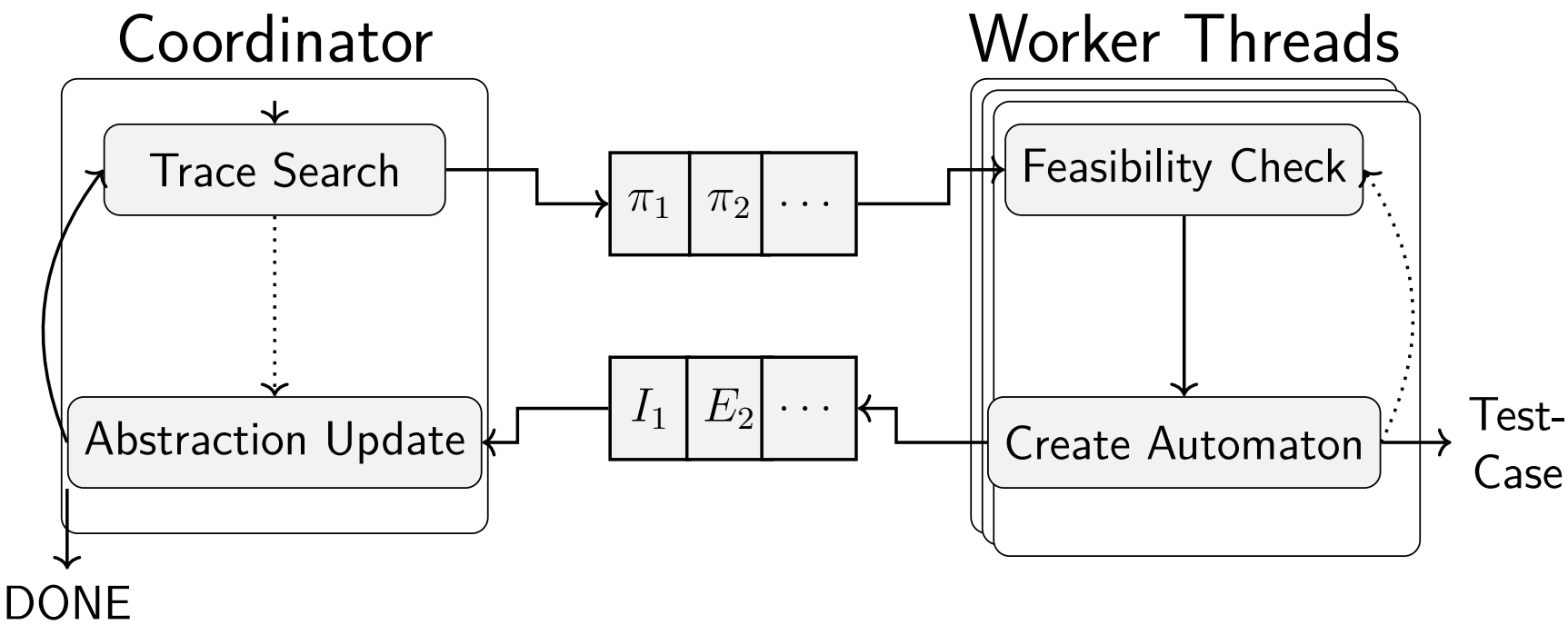
What is New?

- We utilize a new framework for parallel Trace Abstraction
  - Decrease response time
- A prototype for a cooperative / portfolio approach within the framework.
  - Alternative to purely interpolation based trace abstraction.

Overview Sequential UTestGen



Overview Test-Case Generation



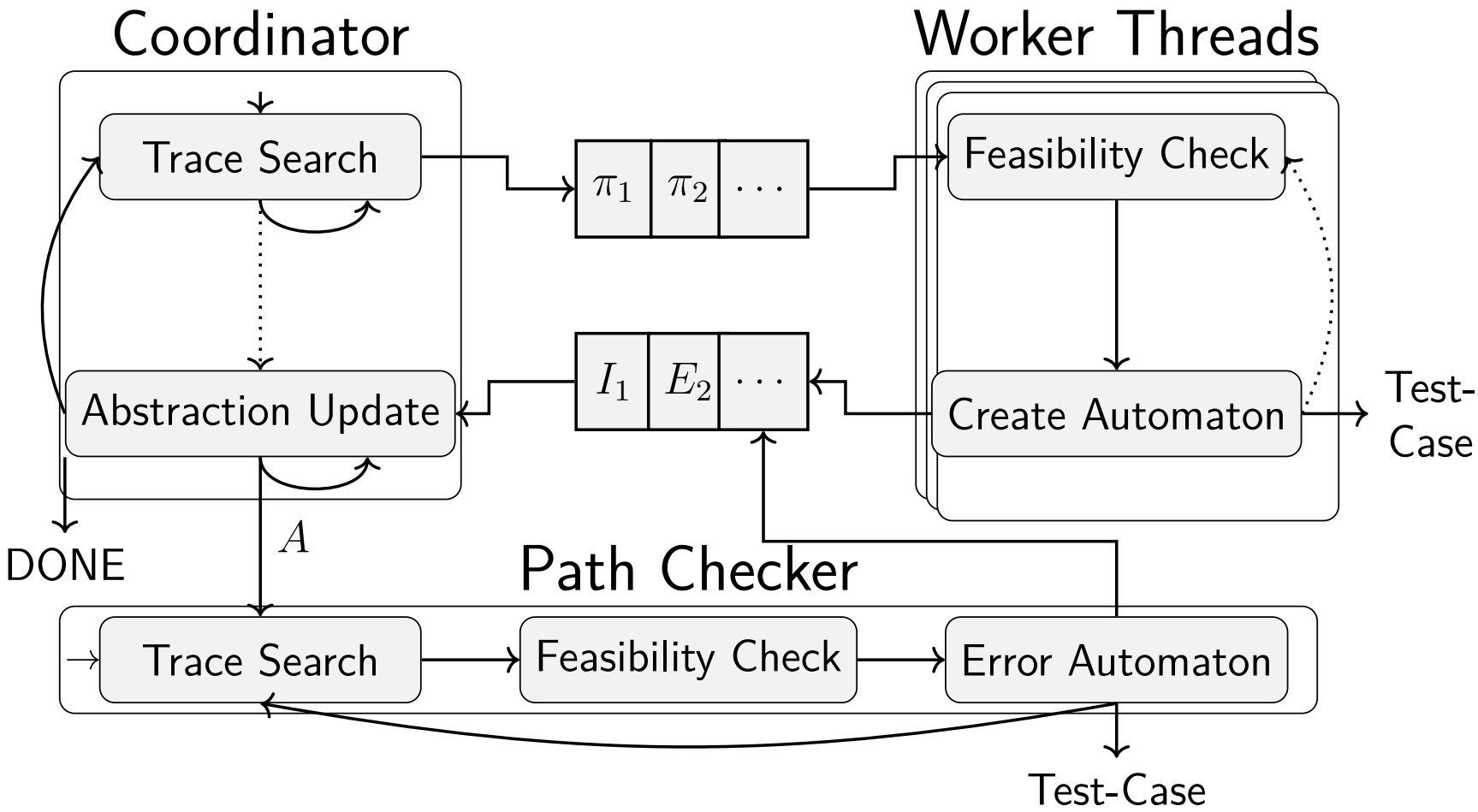
- Two Buffers: Counterexamples and Automata
- Dotted arrow indicates waiting for a Buffer.

Cooperative / Portfolio Prototype

- Trace Abstraction relies heavily on interpolation.
- We often get stuck during interpolation and refinement.
- Other test-goals could be covered easily.

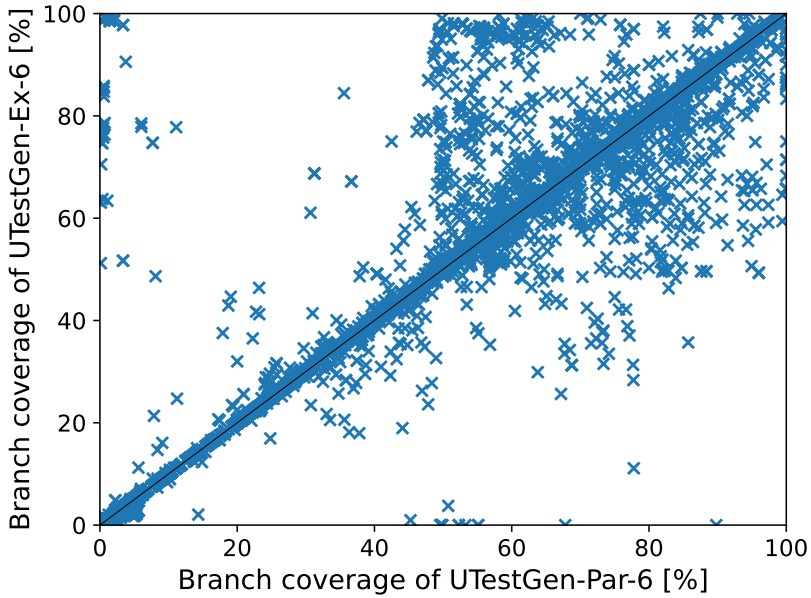
$\Rightarrow$ Cooperation within our framework can help with this issue.

Cooperative / Portfolio Prototype



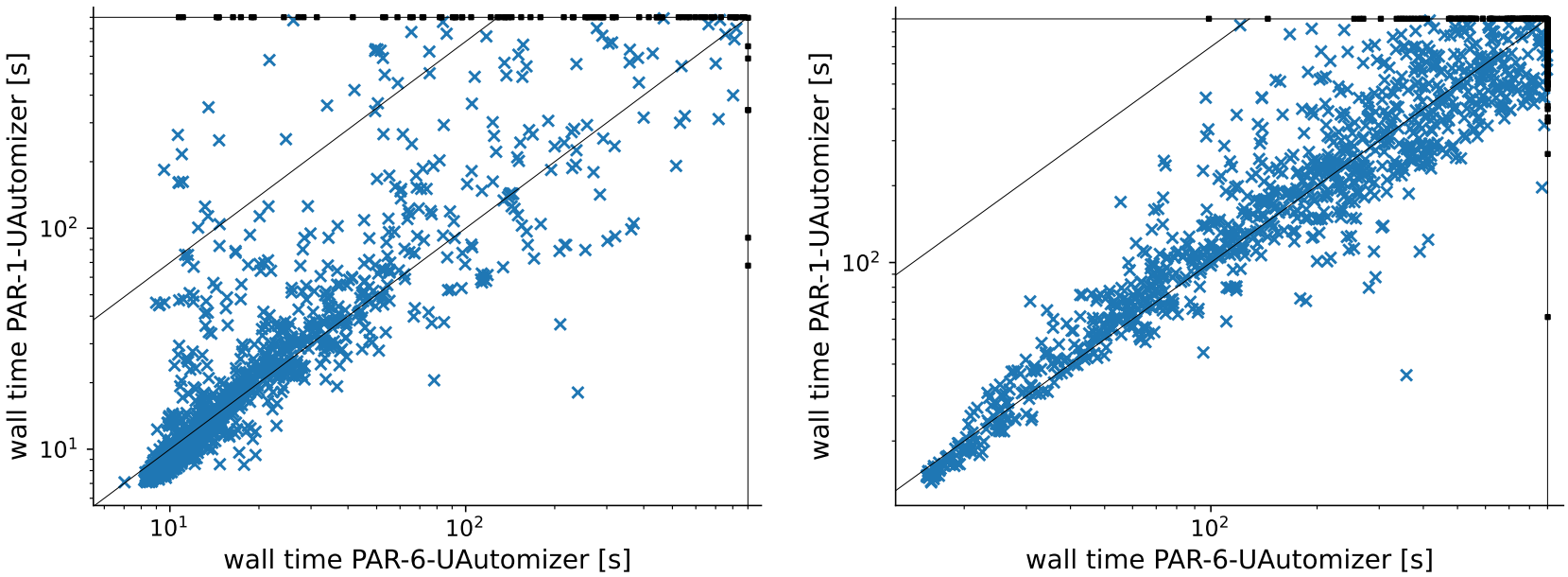
- A naive symbolic execution
  - Keeps infeasible traces in $\mathcal{S}_{\pi_{\text{ex}}}$
  - Runs on the latest abstraction, loops are bound.

Test-case Generation: Impact Path Checker on Coverage



- 6 TA Worker: 533000 Coverage
- 5 TA Worker +1 Path Checker: 541000 Coverage

Test-case Generation: Integer Mode VS. Bit-Vector Mode



- Integer Mode: Linear Integer Arithmetic
- Bit-Vector Mode: Bit-precise reasoning / Floating points

UTestGen at Test-Comp26

- Participated in Cover-Error and Cover-Branches
- We focus on Cover-Branches:
  - Raw coverage of 7500, an increase from last year.
  - The improvement is not reflected in the score.

Future Work:

- Replace the Path-Checker by proper Symbolic Execution on the Abstraction Automaton.