CPA-Daemon:
Mitigating Tool Restarts for Java-based Verifiers

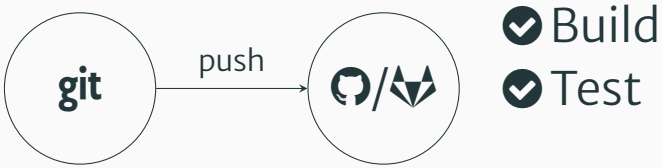
Henrik Wachowitz henrik.wachowitz@ifi.lmu.de
Dirk Beyer, Thomas Lemberger

September 9, 2024
LMU Munich, SoSy-Lab

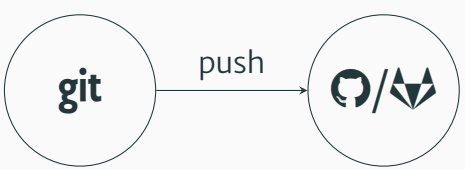
LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

SoSy-Lab Software Systems

Modern Workflows

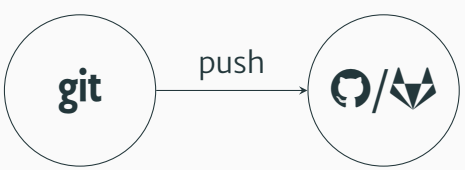


Modern Workflows



- Build
- Test
- Verify?

Modern Workflows

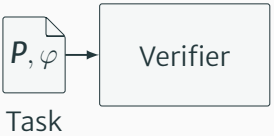


- Build
- Test
- Verify?

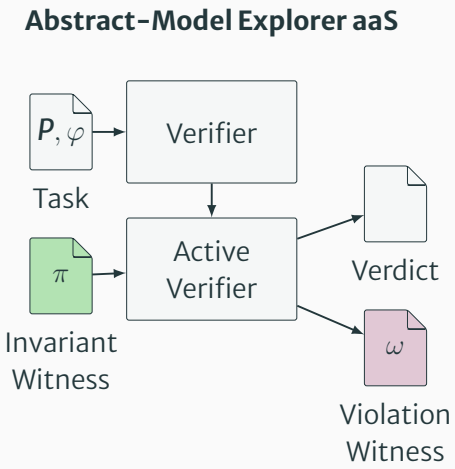
- Challenge: **Machine friendly** interaction.
- Challenge: Fast **response time**.

Key Idea: Verification as a Service

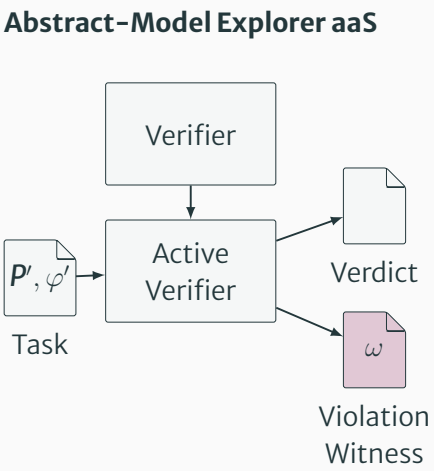
Abstract-Model Explorer aaS



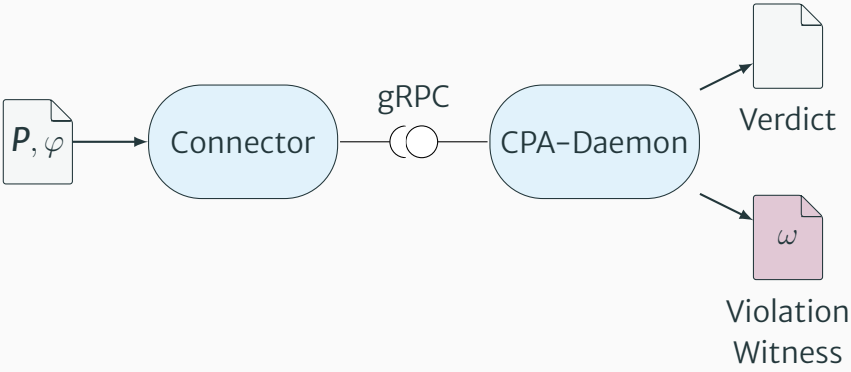
Key Idea: Verification as a Service



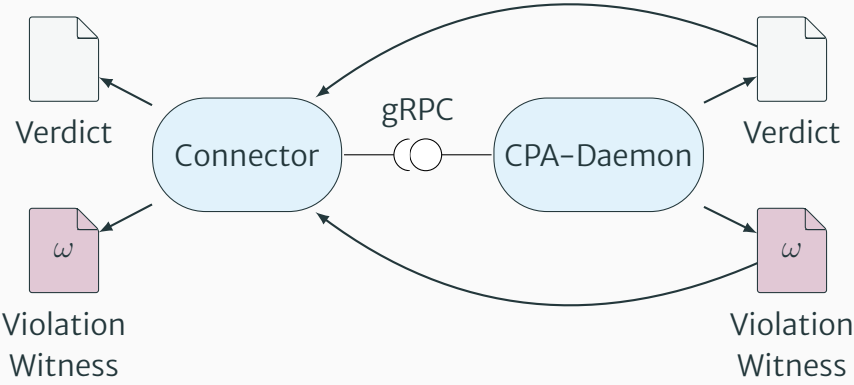
Key Idea: Verification as a Service

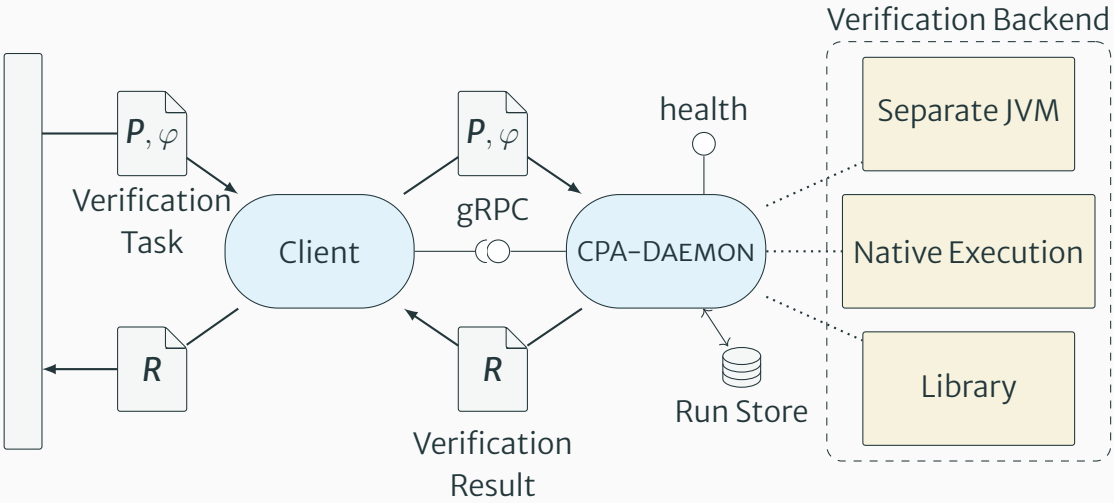


CPA-Daemon: Workflow

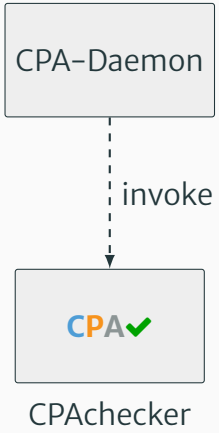


CPA-Daemon: Workflow



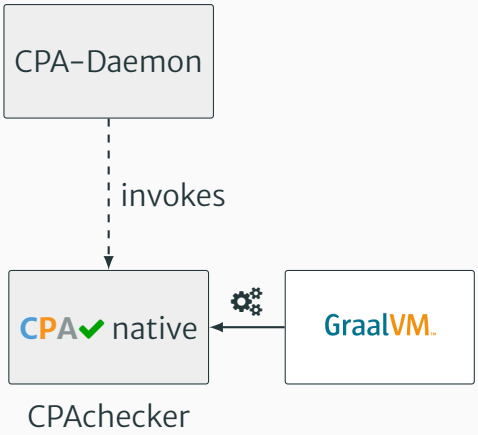

Verification Backend
health
Separate JVM
P, φ
Verification Task
P, φ
gRPC
Client
CPA-DAEMON
Native Execution
R
R
Run Store
Library
Verification Result

CPA-Daemon: Separate JVM



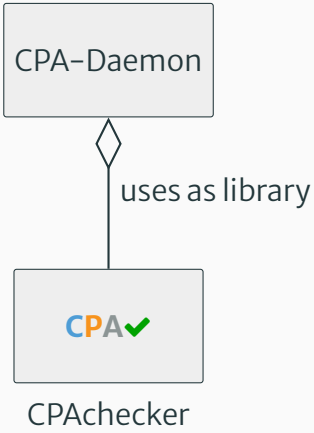
- CPA executed in a fresh JVM
- **Baseline:** Should work just like CPA alone

CPA-Daemon: Native Execution

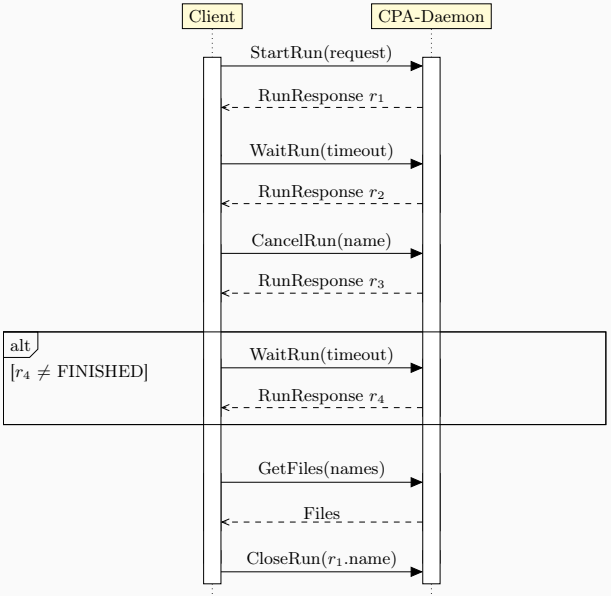


- compile CPA✔ with GraalVM to native binary
- CPA✔ as native ⇒ no JVM need
- ideally: Significant speedup

CPA-Daemon: Library

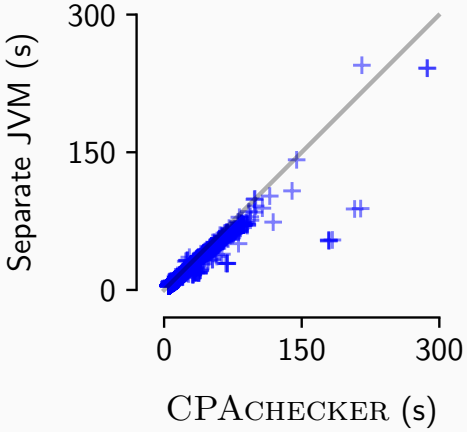


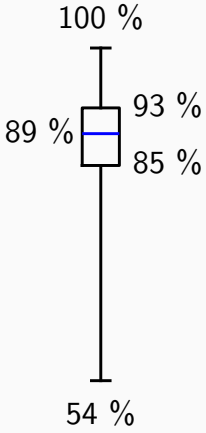
- **CPA-Daemon** is a continuously running service
- CPA✔ executed in the same JVM



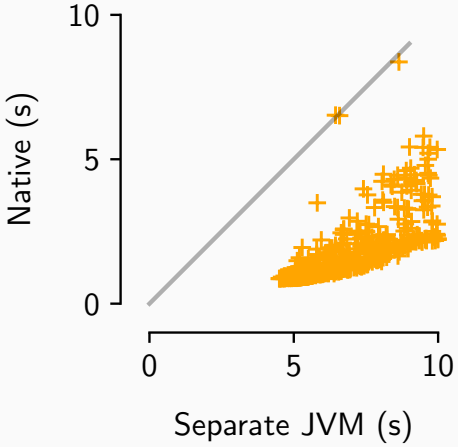
- Python Client serves as connector in Experiments
- Blocks like “normal” CPA✔

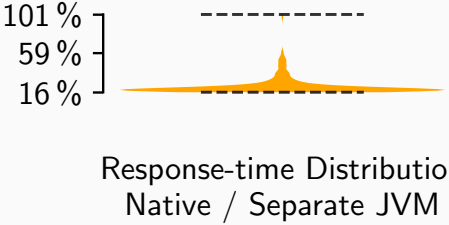
Overhead?



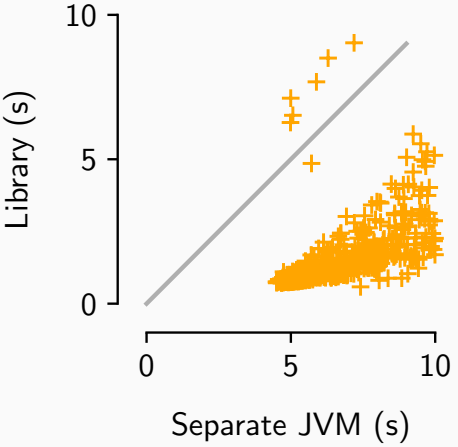


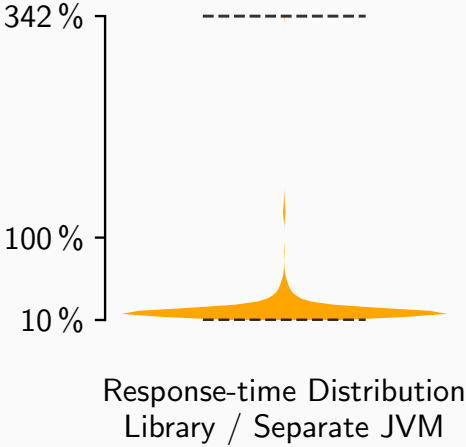
Benefit on Fast-to-solve Tasks



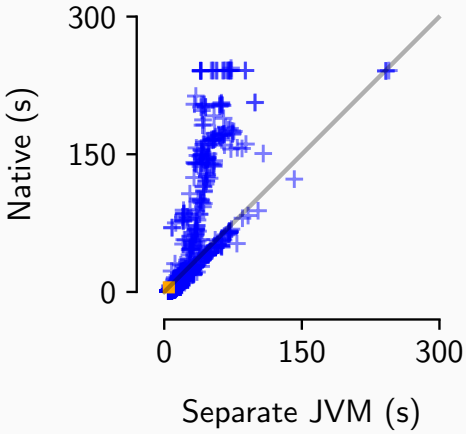


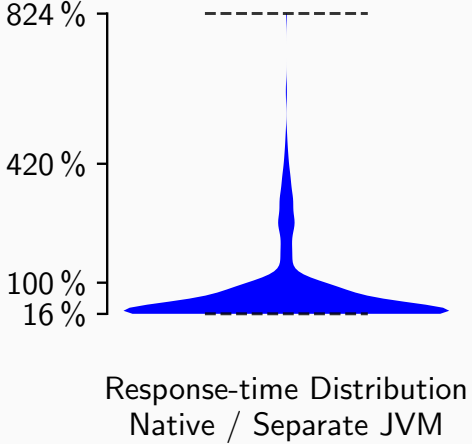
Benefit on Fast-to-solve Tasks



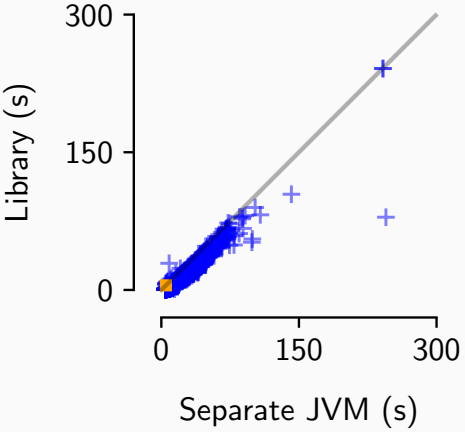


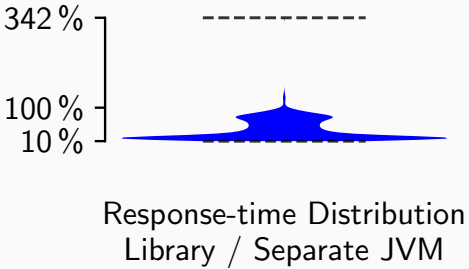
Scalability



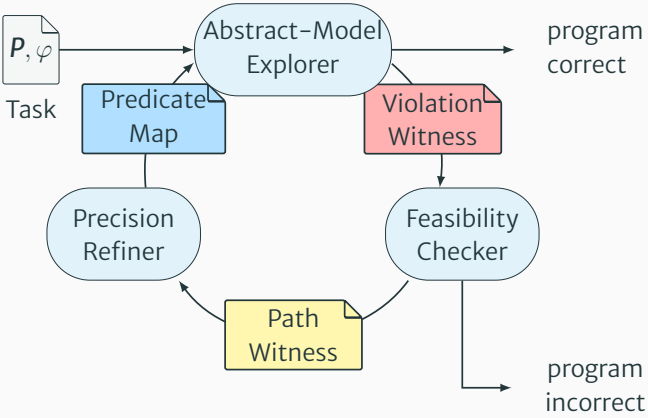


Scalability

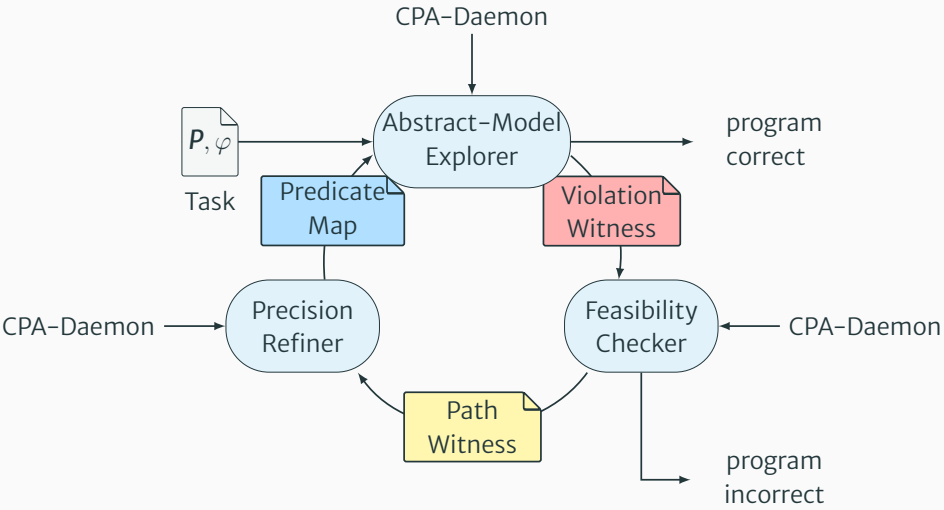




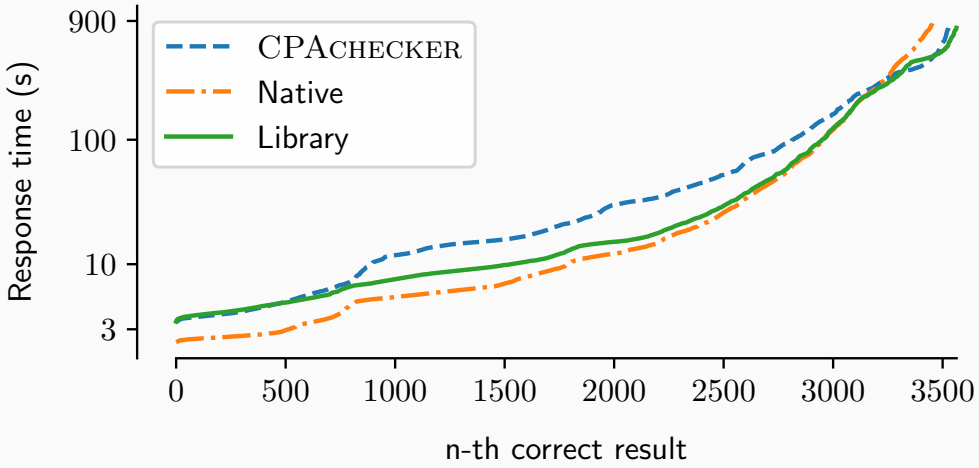
Recall: C-CEGAR



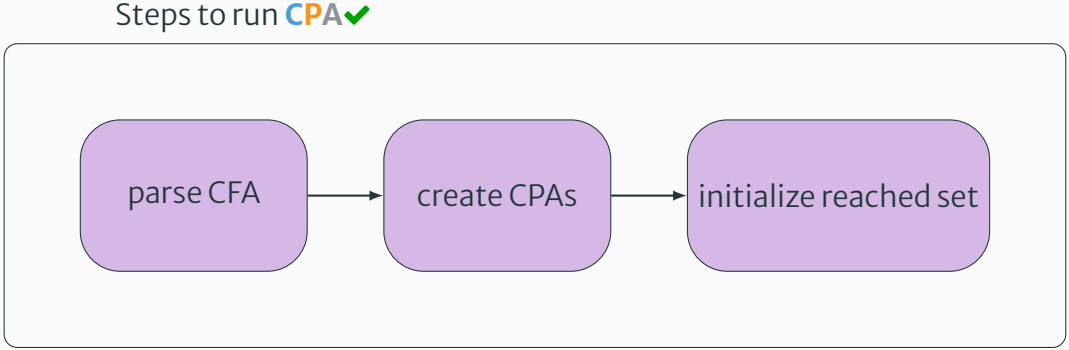
Recall: C-CEGAR



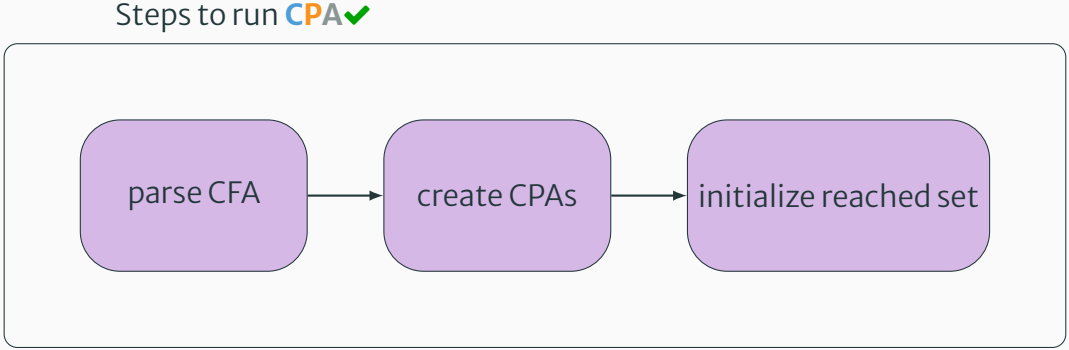
Evaluation


CPAchecker
Native
Library
Response time (s)
900
100
10
3
0
500
1000
1500
2000
2500
3000
3500
n-th correct result

Outlook: Levels of Reuse

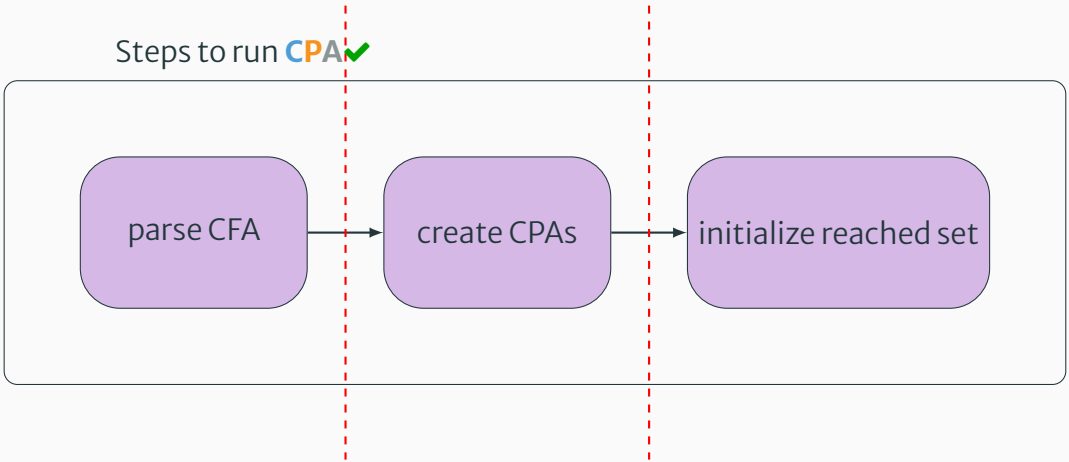


Outlook: Levels of Reuse

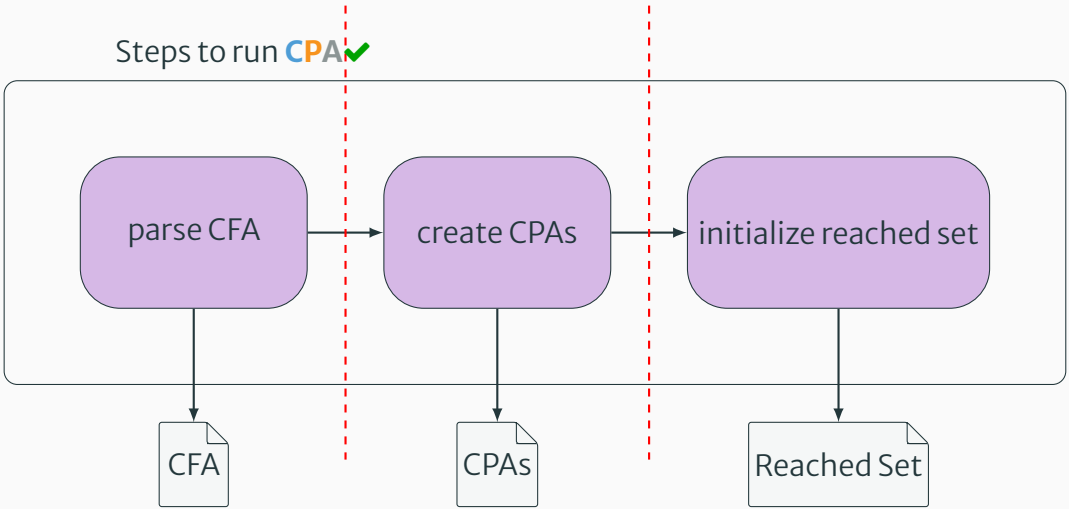


- **Idea**: Reuse artifacts from these steps

Outlook: Levels of Reuse



Outlook: Levels of Reuse



Reusing Artifacts

Stateless represen-
tation: cache it!

Reusing Artifacts

Stateless represen-
tation: cache it!

Alogrithms in theory
stateless: Created
Objects reusable?

Reusing Artifacts

Stateless represen-
tation: cache it!

Alogrithms in theory
stateless: Created
Objects reusable?

Currently too
tightly coupled
with internal state