RELIA: Accelerating the Analysis of Cloud Access Control Policies

Dan Wang[1], Peng Zhang[1], Zhenrong Gu[1], Weibo Lin[2], Shibiao Jiang[2],
Zhu He[2], Xu Du[2], Longfei Chen[2], Jun Li[2], and Xiaohong Guan[1]

XI'AN JIAOTONG UNIVERSITY

HUAWEI

OVERVIEW

- Cloud providers offer flexible access control by **access control policies** to secure user's cloud resources. However, configuring access control policies is **error-prone**. Cloud providers have developed SMT-based tools to formally analyze the user-defined policies.
- Unfortunately, these analyzers are **slow**, due to the **complex regular expression** matching conditions in policies.
- We propose **RELIA**, a general method to speed up the analysis of cloud access control policies. RELIA **pre-computes** a set of ***String Equivalence Classes (SECs)*** based on the regular expressions in a policy, assign a unique **integer** to each SEC, and **rewrite** the regular constraints into equivalent **integer** constraints, which are easier to solve. We implement RELIA as a **transparent layer** between policy analyzer and off-the-shelf SMT solvers.
- Based on **real policies** from Huawei, we show that: when enabling RELIA, our in-house portfolio solver can **speed up** the analysis process for nearly **95%** of all cases, with an average speedup of **8.21×**.

BACKGROUND

As cloud services thrive, resources on cloud need to be secured. Many companies outsources their services to the cloud. To secure those assets, cloud service providers (CSPs) enable users to configure access control policies. However, these policies is complex and error-prone. A misconfiguration in the policies can lead to security risks, such as exposing private resources to the public, or allowing attackers to gain unauthorized access by escalating privileges.

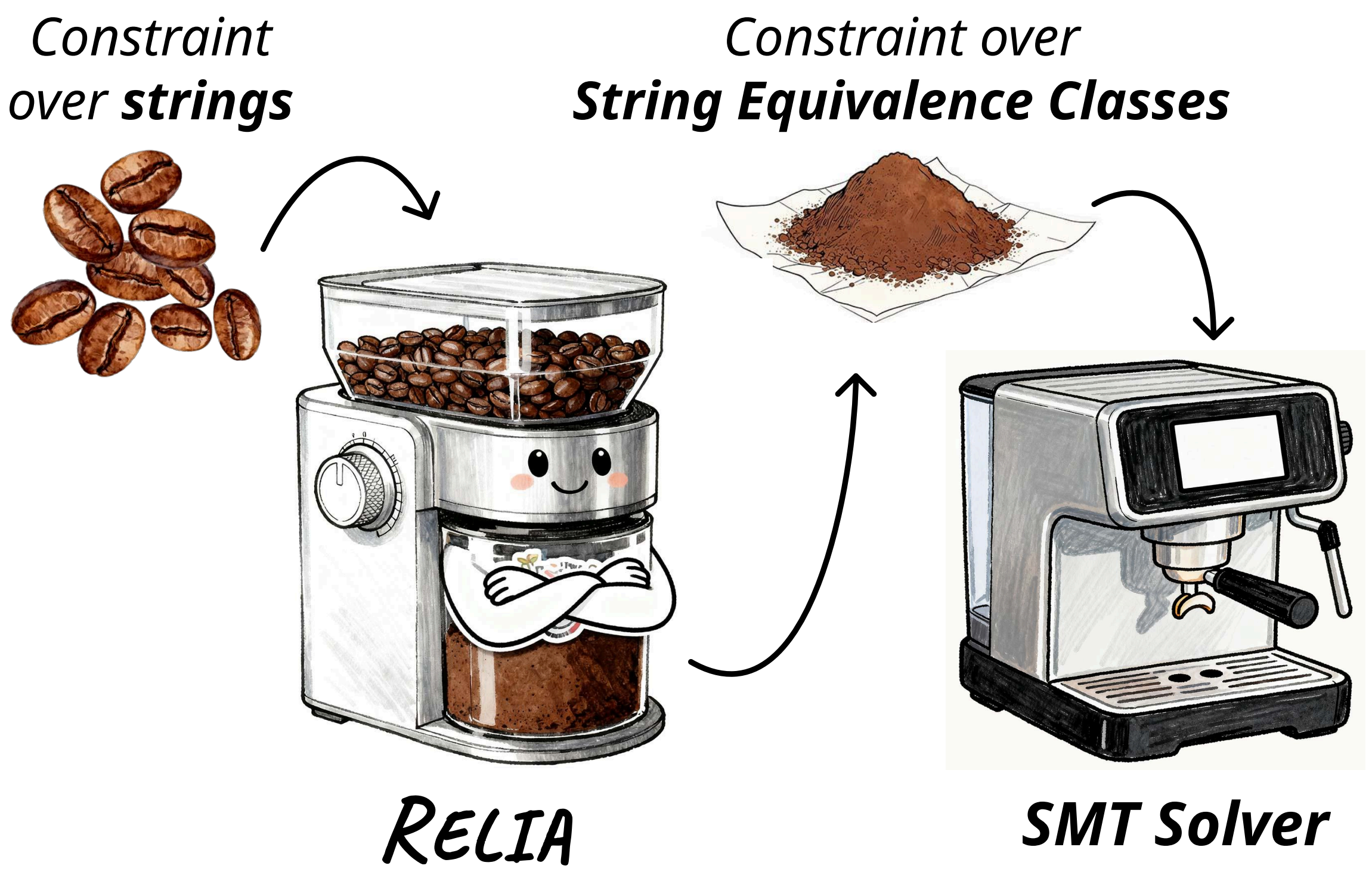
SMT-Based access analyzers are slow. Evaluation results show that general-purpose SMT solvers (e.g., Z3, CVC4, and CVC5) are slow when analyzing some complex policies.

String search space is huge, slowing down SMT solving. However, a lot of strings satisfy the same set of regular constraints and thus can be merged into an equivalence class *(or string equivalence class, SEC)*. We found that the total number of SECs is small for cloud access control policies: on 506 real policies from Huawei Cloud, the total number of SECs is always less than 12.

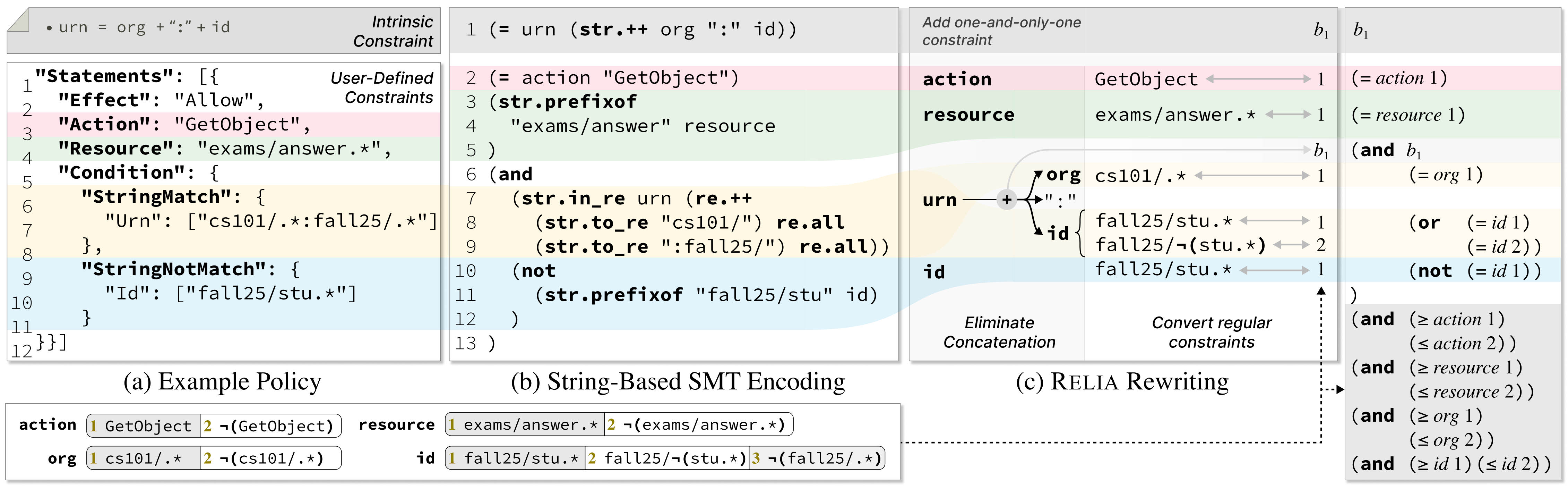
We propose **RELIA** to utilize the reduced number of SECs and accelerate string solving in policy analysis.

KEY IDEAS

RELIA converts constraints over strings to constraints over *String Equivalence Classes*!



1. **Eliminate concatenations.** ACP often contains *compound string variables*, e.g. urn *(uniform resource name)* = org *(organization)* + ":" + id. If there is a regular expression constraint on it, it also constrains the component variables, i.e. org and id. RELIA distributes the constraints to its component constraints to achieve equisatisfiability.
2. **Convert regular constraints.** RELIA converts regular constraints of string variables into linear integer arithmetic (LIA) constraints over integer variables, base on which SECs satisfies the original constraint.



```
• urn = org + ":" + id
```

Intrinsic Constraint

User-Defined Constraints

```
"Statements": [{
  "Effect": "Allow",
  "Action": "GetObject",
  "Resource": "exams/answer.*",
  "Condition": {
    "StringMatch": {
      "Urn": ["cs101/.*:fall25/.*"]
    },
    "StringNotMatch": {
      "Id": ["fall25/stu.*"]
    }
}}]
```

(a) Example Policy

```
(= urn (str.++ org ":" id))

(= action "GetObject")
(str.prefixof
  "exams/answer" resource
)
(and
  (str.in_re urn (re.++
    (str.to_re "cs101/") re.all
    (str.to_re ":fall25/") re.all))
  (not
    (str.prefixof "fall25/stu" id)
  )
)
```

(b) String-Based SMT Encoding

(c) RELIA Rewriting

```
b1
(= action 1)
(= resource 1)
(and b1
      (= org 1)

      (or  (= id 1)
           (= id 2))
      (not (= id 1))
)
(and (≥ action 1)
     (≤ action 2))
(and (≥ resource 1)
     (≤ resource 2))
(and (≥ org 1)
     (≤ org 2))
(and (≥ id 1) (≤ id 2))
```

(d) LIA-Based SMT Encoding

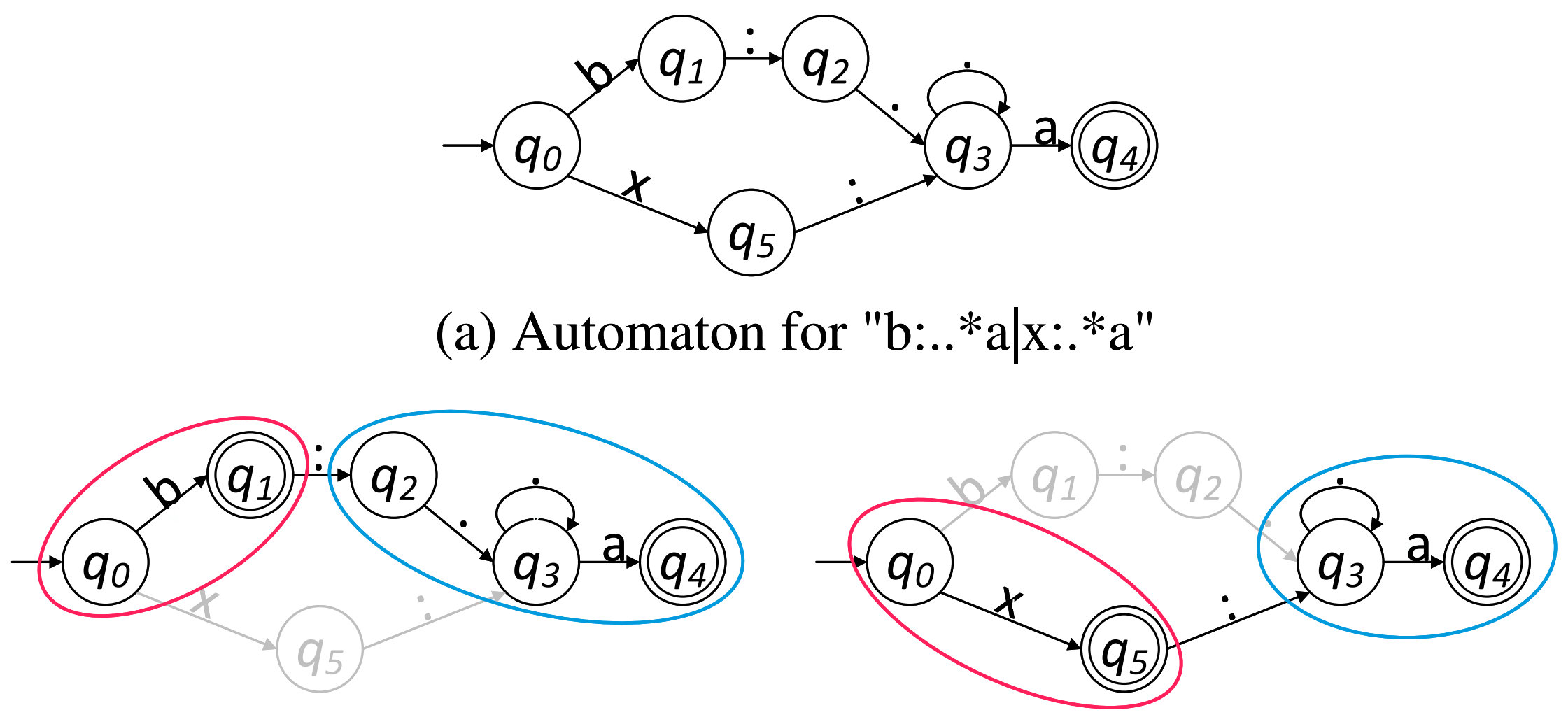
action	1 GetObject	2 ¬(GetObject)	
org	1 cs101/.*	2 ¬(cs101/.*)	
resource	1 exams/answer.*	2 ¬(exams/answer.*)	
id	1 fall25/stu.*	2 fall25/¬(stu.*)	3 ¬(fall25/.*)

(e) String Equivalence Class (SEC)

HANDLING STRING CONCATENATIONS

Compound string variables in ACP have easy forms. For clarity, compound string variables in ACP have:

- **Anchor points**. The component variables of a compound string variable are separated by string literals like ":" or "/", and they should not appear in these variables to eliminate ambiguity.
- **One-and-only-one concatenation**. A compound variable can have multiple concatenation schemes (such as urn can be org + ":user:" + user_id for individuals, or org + ":agency:" + session_id for agencies); but one and only one concatenation scheme is valid given a value of the variable.



(a) Automaton for "b:..*a|x:.*a"

(b) Split Case 1

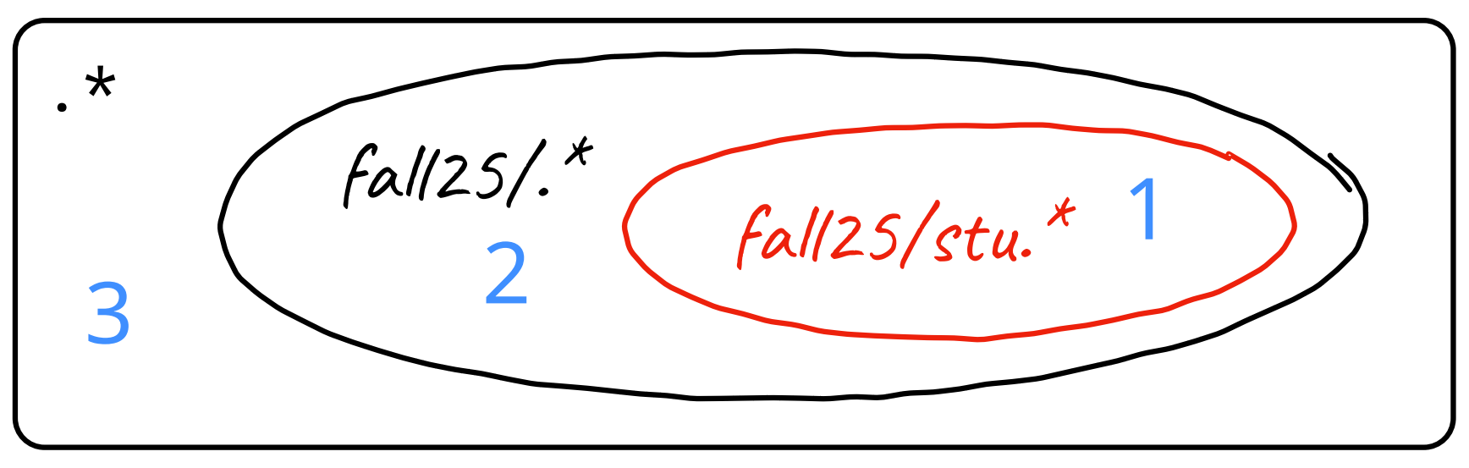
(c) Split Case 2

We can split regular constraints easily on anchor points of their automata. If compound variable s have regular constraint $s \in R$, it has component variables s_0, s_1, $\cdots$, s_m, and having N split cases, It can be converted into:

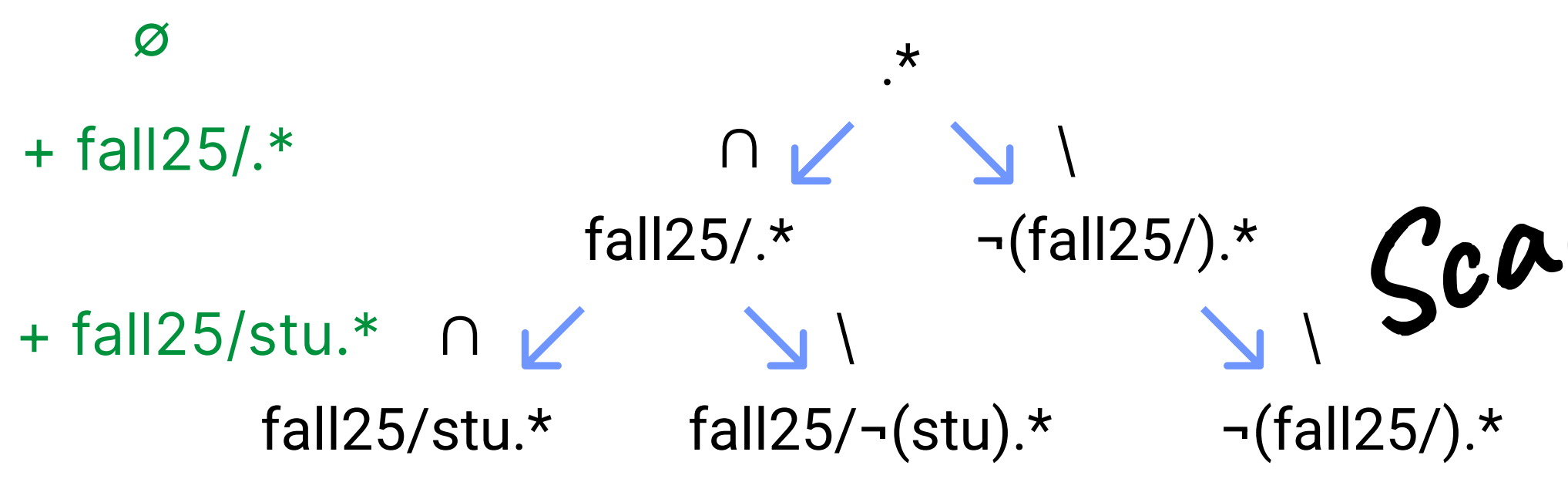
$$s \in R \Longrightarrow \bigvee_{i\in[1,N]} \left(\bigwedge_{j\in[1,m]} s_j \in R_i^j \right)$$

CONVERTING REGULAR CONSTRAINTS

Computes SECs. RELIA computes the SECs regarding to all the regular constraints of each variable. If a variable has 2 regular constraints: fall25/.* and fall25/stu/.*, they divide the string space into 3 SECs, labeled:



SEC is computed by iterate through regular constraints and compute FA / set intersection and difference with previous SECs.



∅

+ fall25/.*

.*

∩ ↙ ↘ \

fall25/.* ¬(fall25/).*

+ fall25/stu.* ∩ ↙ ↘ \ ↘ \

fall25/stu.* fall25/¬(stu).* ¬(fall25/).*

Finally turn regular constraints into LIA constraints. If $s \in R$ *iff* s is in SECs labeled $I_R = \{i_1, i_2, \cdots i_N\}$, then s can be represented with integer variable i_s, and $s \in R$ is converted into:

$$s \in R \Longrightarrow \bigvee_{n\in I_R} i_s = n$$

EVALUATIONS

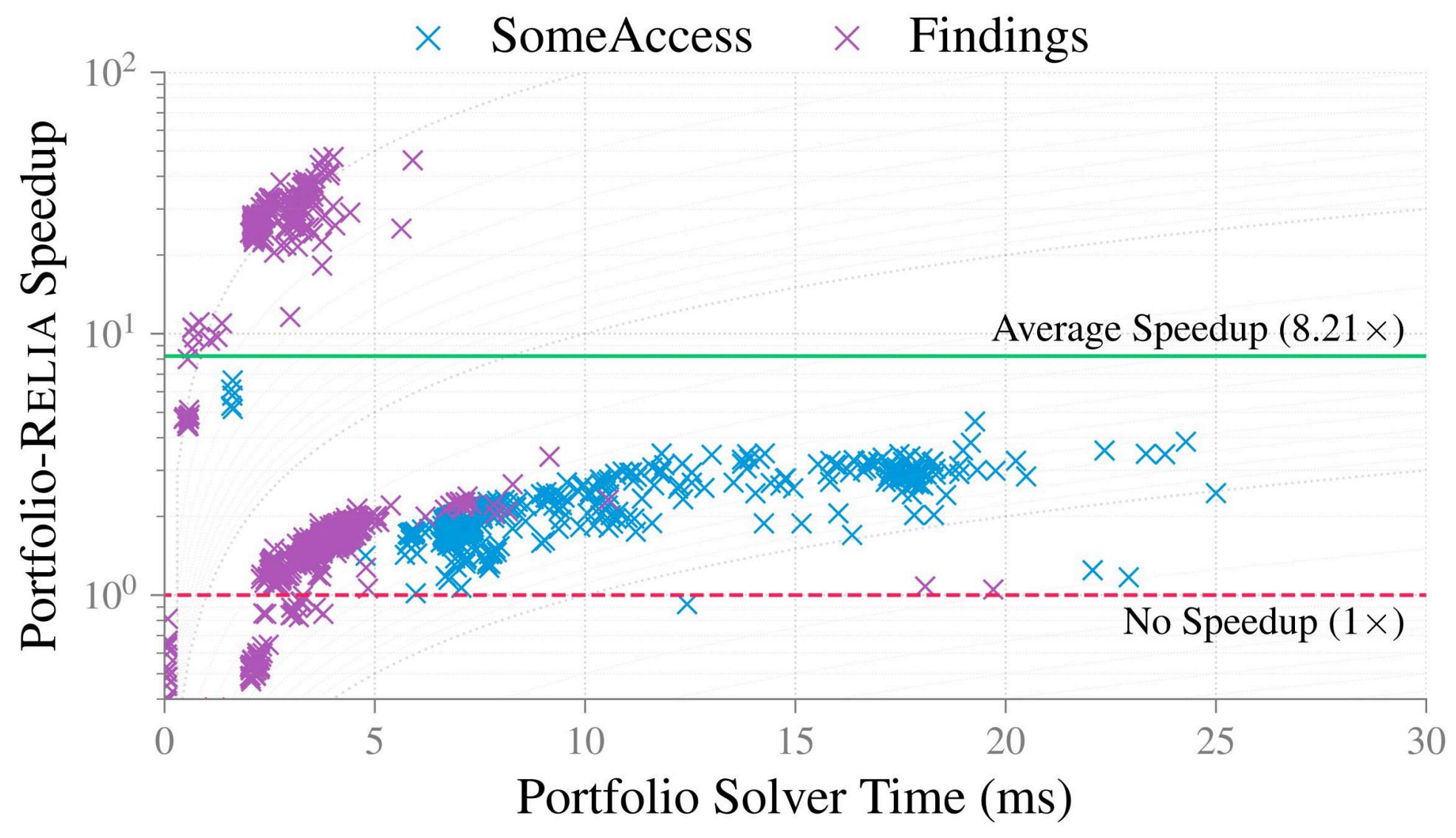
RELIA is able to:

☑ Accelerate **real** policy analysis by **8.21**×;

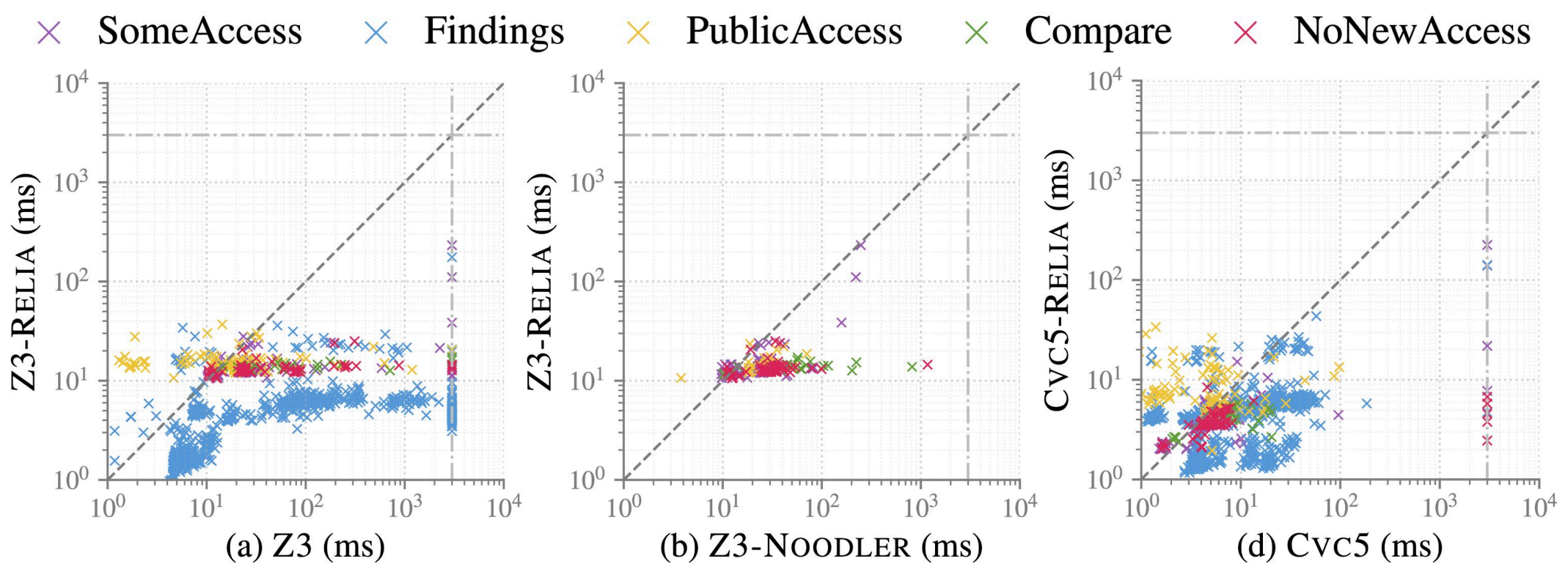
☑ Analyze **8/10** harder policies in 3s which Z3, Cvc4 and Cvc5 cannot solve most in hours;

☑ Generalize to and accelerate **other CSP**'s ACP analysis;

☑ Solve **general** string SMT problems at great speed.



Speedup of real Huawei policy analysis

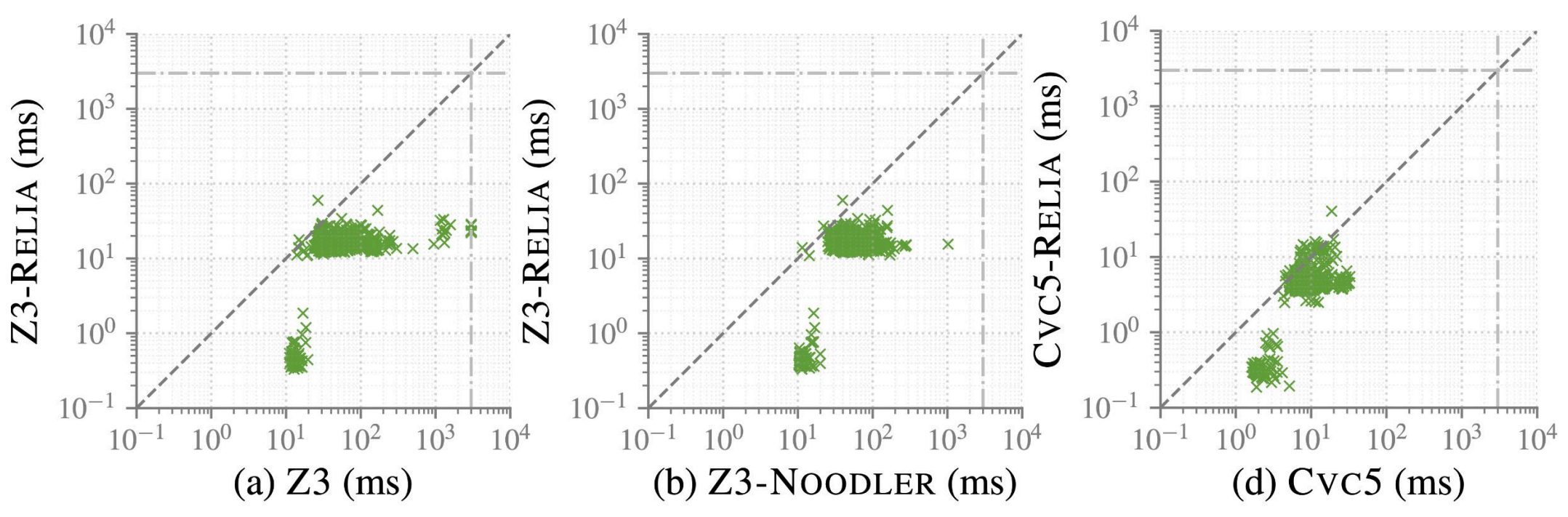


(a) Z3 (ms)

(b) Z3-NOODLER (ms)

(d) CVC5 (ms)

Speedup of difference analyses types

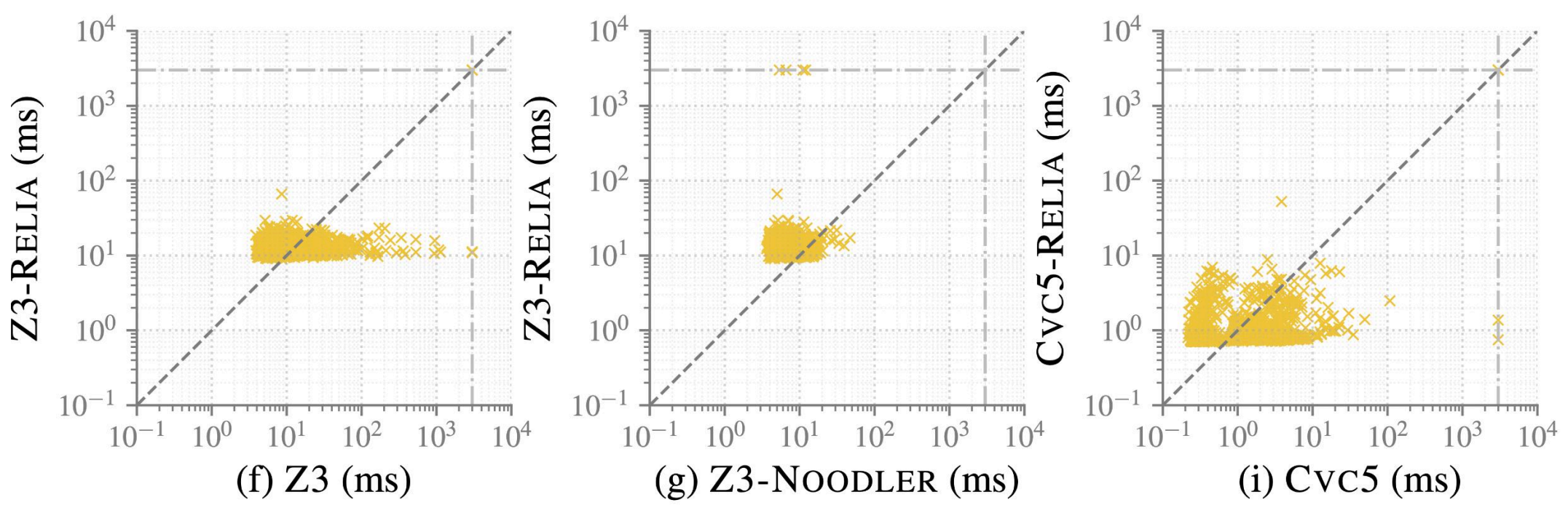


(a) Z3 (ms)

(b) Z3-NOODLER (ms)

(d) CVC5 (ms)

Speedup of translated AWS policies from Quacky



(f) Z3 (ms)

(g) Z3-NOODLER (ms)

(i) CVC5 (ms)

Speedup of general string SMT problems from AutomatArk

Scan the QR Codes for more!

Presentation

Paper

ASE 2025 THE 40TH IEEE/ACM INTERNATIONAL CONFERENCE ON AUTOMATED SOFTWARE ENGINEERING
NOV 16 - 20, 2025
SEOUL, KOREA