1 西安交通大學 XI'AN JIAOTONG UNIVERSITY 2 HUAWEI

AccessRefinery: Fast Mining Concise Access Control Intents on Public Cloud

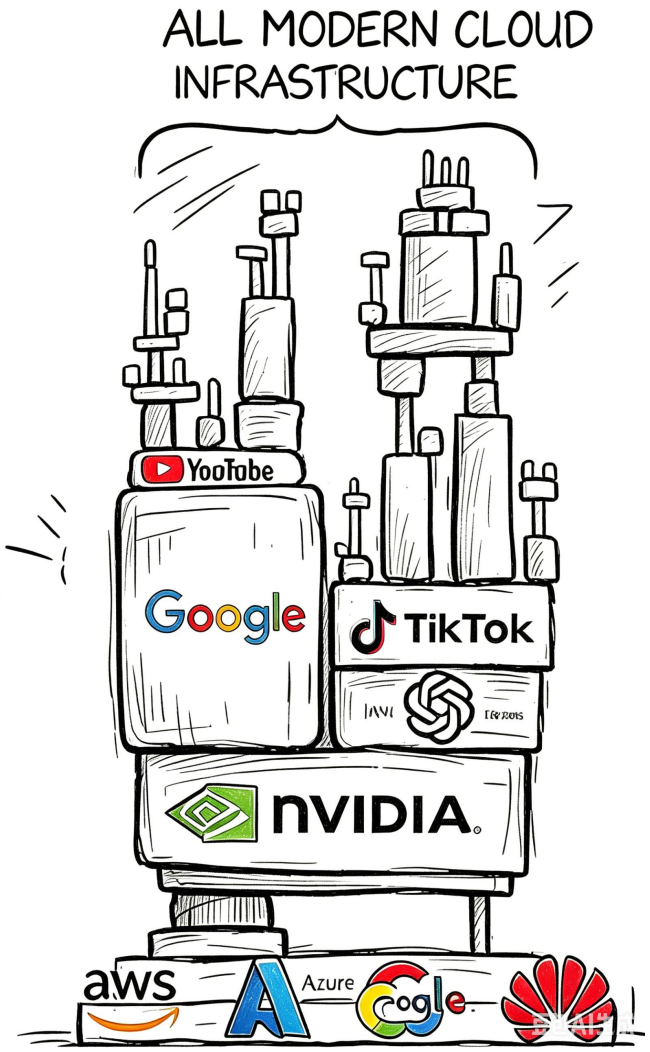
[1]**Ning Kang**, [1]Peng Zhang, [1]Jianyuan Zhang, [1]Hao Li, [1]Dan Wang, [1]Zhenrong Gu,
[2]WeiBo Lin, [2]Shibiao Jiang, [2]Zhu He, [2]Xu Du, [2]Longfei Chen, [2]Jun Li, [1]Xiaohong Guan

The ACM International Conference on the Foundations of Software Engineering (FSE)

Montreal, Canada
Sun 5 - Thu 9 July 2026

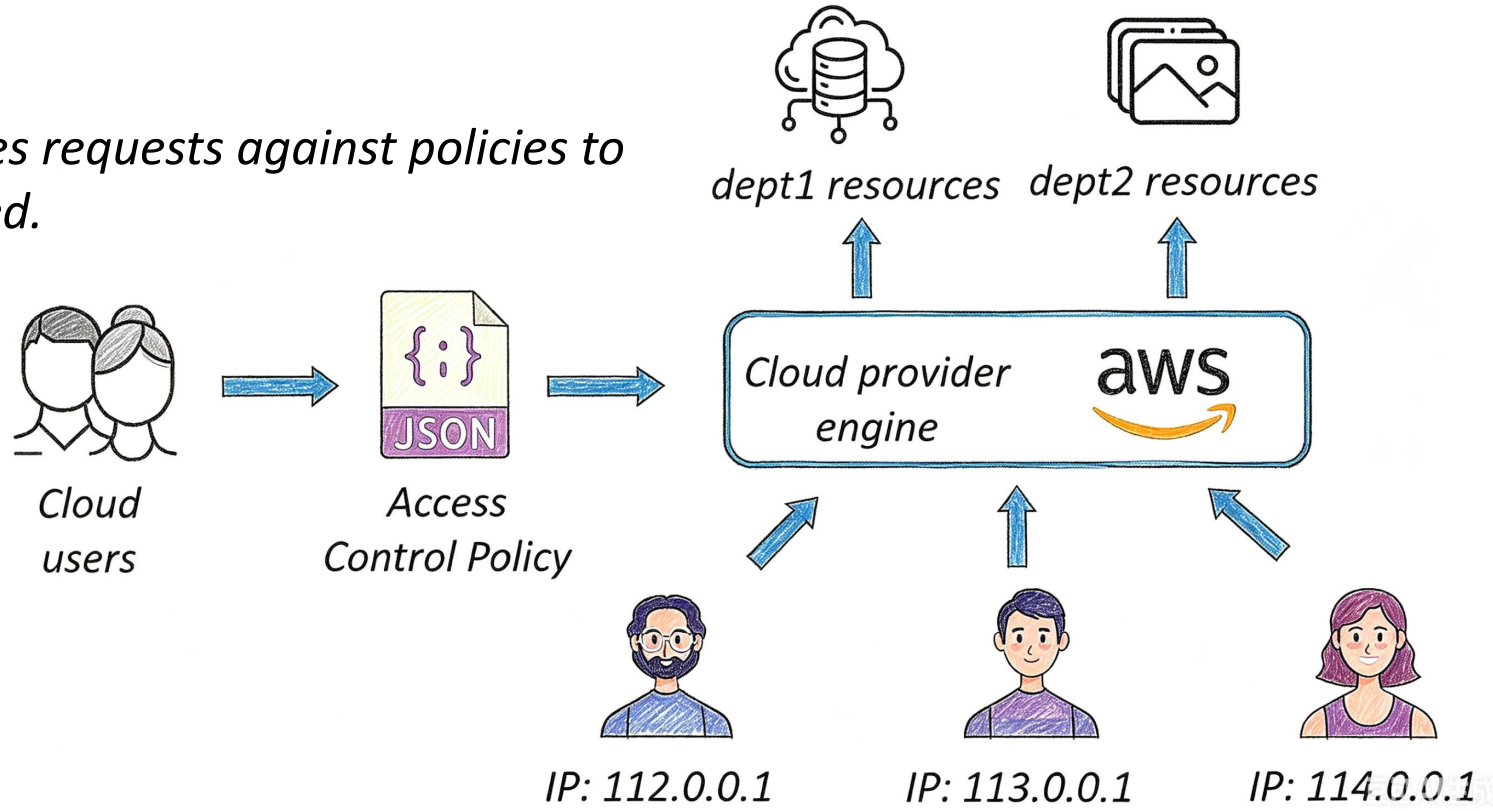
Background

- IAM plays an important role in cloud security.
  - *Software applications are increasingly deployed **in the cloud.***
  - *To secure these applications, major cloud providers offer Identity and Access Management **(IAM) systems.***

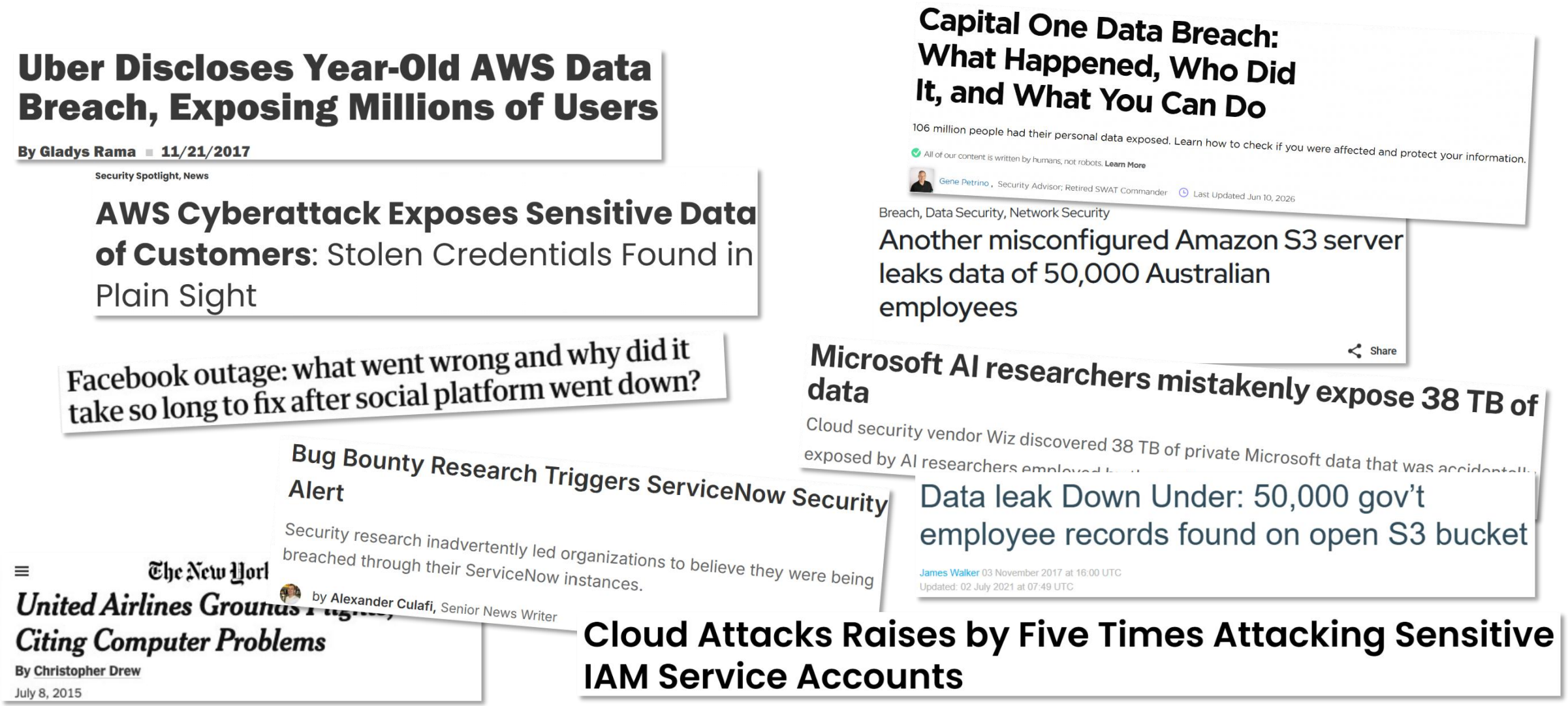


What is IAM ?

- IAM systems follow a shared responsibility model.
  - *The **cloud users** secure their services by writing access control policies (or IAM policies).*
  - *The **cloud provider engine** evaluates requests against policies to determine whether access is allowed.*



IAM policies are error-prone



Why are IAM policies error-prone?

- Wildcard semantics:
  - *dept1/user*.txt matches any file in the dept1 directory whose name starts with user and ends with .txt*
  - *112.0.0.0/24 restricts the range from 112.0.0.0 to 112.0.0.255*

Statement1	*Effect*	: *Allow*
	Resource	: *dept*/user1.txt, dept1/user*.txt*
	IpAddress	: *112.0.0.0/24, 113.0.0.0/24*
Statement2	*Effect*	: *Deny*
	NotResource	: *dept*/user1.txt*
	IpAddress	: *112.0.0.0/24*
Statement3	*Effect*	: *Deny*
	NotResource	: *dept1/user*.txt*
	IpAddress	: *113.0.0.0/24*

IAM Policy

Why are IAM policies error-prone?

- Wildcard semantics:
  - *dept1/user*.txt matches any file in the dept1 directory whose name starts with user and ends with .txt*
  - *112.0.0.0/24 restricts the range from 112.0.0.0 to 112.0.0.255*
- Boolean semantics:
  - *With one domain (key), different values have OR relationship.*
  - *Between different keys, the relationship is AND.*
  - *NOT can be indicated by adding Not before the domain.*

Statement1	*Effect*	: *Allow*
	Resource	: *dept*/user1.txt, dept1/user*.txt*
	IpAddress	: *112.0.0.0/24, 113.0.0.0/24*
Statement2	*Effect*	: *Deny*
	NotResource	: *dept*/user1.txt*
	IpAddress	: *112.0.0.0/24*
Statement3	*Effect*	: *Deny*
	NotResource	: *dept1/user*.txt*
	IpAddress	: *113.0.0.0/24*

IAM Policy

Why are IAM policies error-prone?

- Wildcard semantics:
  - *dept1/user*.txt matches any file in the dept1 directory whose name starts with user and ends with .txt*
  - *112.0.0.0/24 restricts the range from 112.0.0.0 to 112.0.0.255*

Statement1

Effect	:	*Allow*
Resource	:	*dept*/user1.txt, dept1/user*.txt*
IpAddress	:	*112.0.0.0/24, 113.0.0.0/24*

- Boolean semantics:
  - *With one domain (key), different values have OR relationship.*
  - *Between different keys, the relationship is AND.*
  - *NOT can be indicated by adding Not before the domain.*

Statement2

Effect	:	*Deny*
NotResource	:	*dept*/user1.txt*
IpAddress	:	*112.0.0.0/24*

Statement3

Effect	:	*Deny*
NotResource	:	*dept1/user*.txt*
IpAddress	:	*113.0.0.0/24*

It is difficult to determine which requests are permitted by a policy.

Request1 is allowed?
Principal : user1
Resource : dept1 / user1 .txt
IpAddress : 112.0.0.32

Request2 is denied?
Principal : user1
Resource : dept1 / user2 .txt
IpAddress : 112.0.0.32

Why are IAM policies error-prone?

- *Wildcard semant*
  - dept1/user*.txt
    name starts wit
  - 112.0.0.0/24 res
- *Boolean semanti*
  - With one domai
  - Between differe
  - NOT can be indi

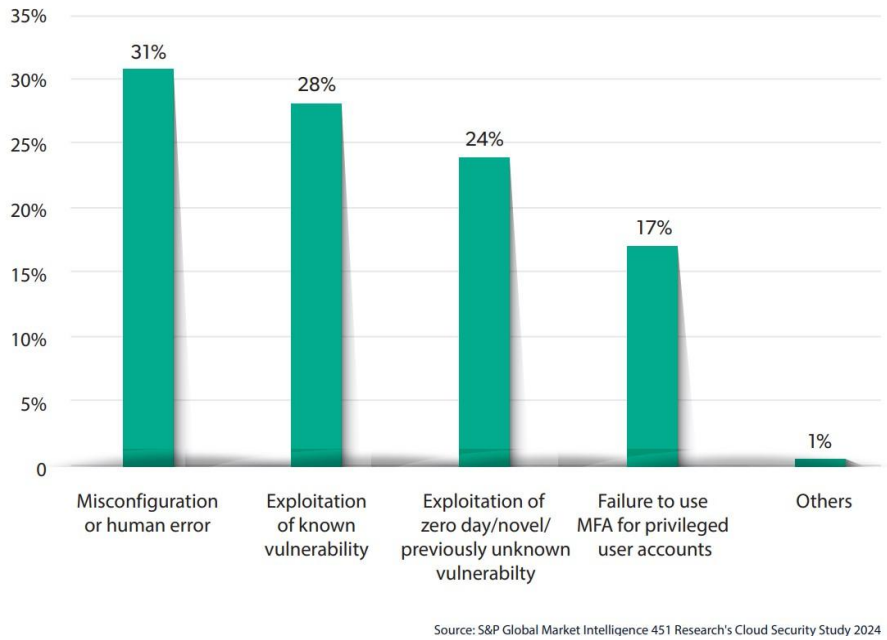
It is difficult to de
requests are perr

Cloud Security: Despite All the Tech, It's Still a People Problem

By David Ramel | 07/02/2024

THALES

Root cause of cloud data breaches



Source: S&P Global Market Intelligence 451 Research's Cloud Security Study 2024

*Credit: https://virtualizationreview.com/articles/2024/07/02/cloud-security.aspx?utm_source=chatgpt.com

ect : Allow
ource : dept*/user1.txt, dept1/user*.txt
ddress : 112.0.0.0/24, 113.0.0.0/24

ect : Deny
tResource : dept*/user1.txt
ddress : 112.0.0.0/24

ect : Deny
tResource : dept1/user*.txt
ddress : 113.0.0.0/24

uest2 is denied?
cipal : user1
ource : dept1 / user2 .txt
ddress : 112.0.0.32

Related Works

- Checking Equivalence or Public Access

  Requires manual definition of an ideal policy or public principle.

Zelkova
FMCAD'18

Ambit
FMCAD'18

Block Public Access
FSE'20

Detecting Attack
Security'23

The ideal policy might be misconfiguration.

Related Works

- Checking Equivalence or Public Access

  Requires manual definition of an ideal policy or public principle.

- Quantifying Allowed Requests

  Requires prior knowledge of the number of allowed requests.

Zelkova
FMCAD'18

Ambit
FMCAD'18

Block Public Access
FSE'20

Detecting Attack
Security'23

Quacky
ASE'22

Projective Model for IP
FMCAD'24

How to understand the number of allowed request is 2^{104}?

Related Works

- Checking Equivalence or Public Access

  Requires manual definition of an ideal policy or public principle.

- Quantifying Allowed Requests

  Requires prior knowledge of the number of allowed requests.

- Mining Access Control Intents

  Help user to understand their policies

Zelkova
FMCAD'18

Ambit
FMCAD'18

Block Public Access
FSE'20

Detecting Attack
Security'23

Quacky
ASE'22

Projective Model for IP
FMCAD'24

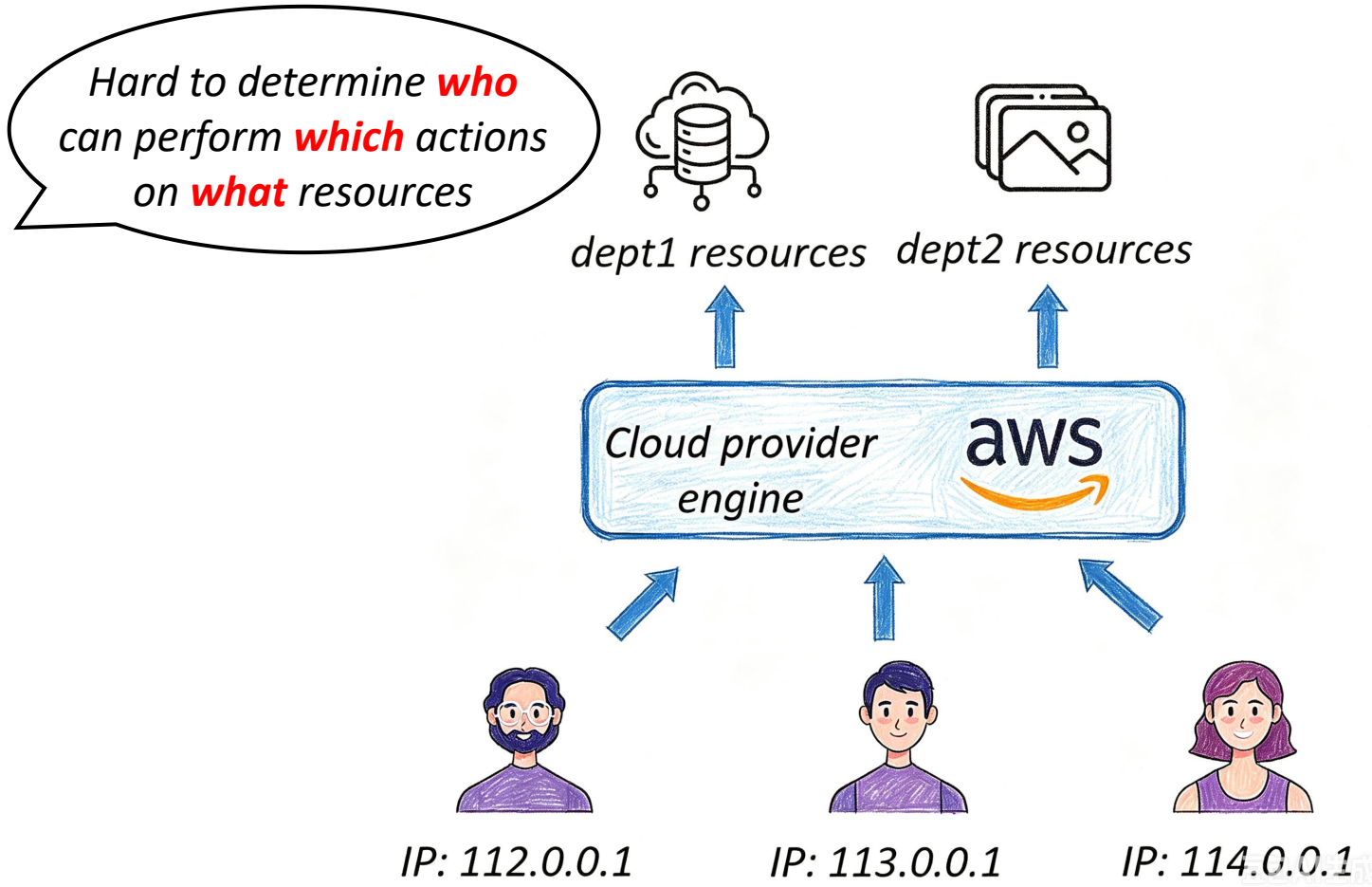
Access Analyzer
CAV'24

How to understand the number of allowed request is 2^{104}?

Intent mining can help users

- Access Control Policies

Statement1	*Effect*	:	*Allow*
	Resource	:	*dept*/user1.txt, dept1/user*.txt*
	IpAddress	:	*112.0.0.0/24, 113.0.0.0/24*
Statement2	*Effect*	:	*Deny*
	NotResource	:	*dept*/user1.txt*
	IpAddress	:	*112.0.0.0/24*
Statement3	*Effect*	:	*Deny*
	NotRes	:	*dept1/user*.txt*
	IpAddress	:	*113.0.0.0/24*



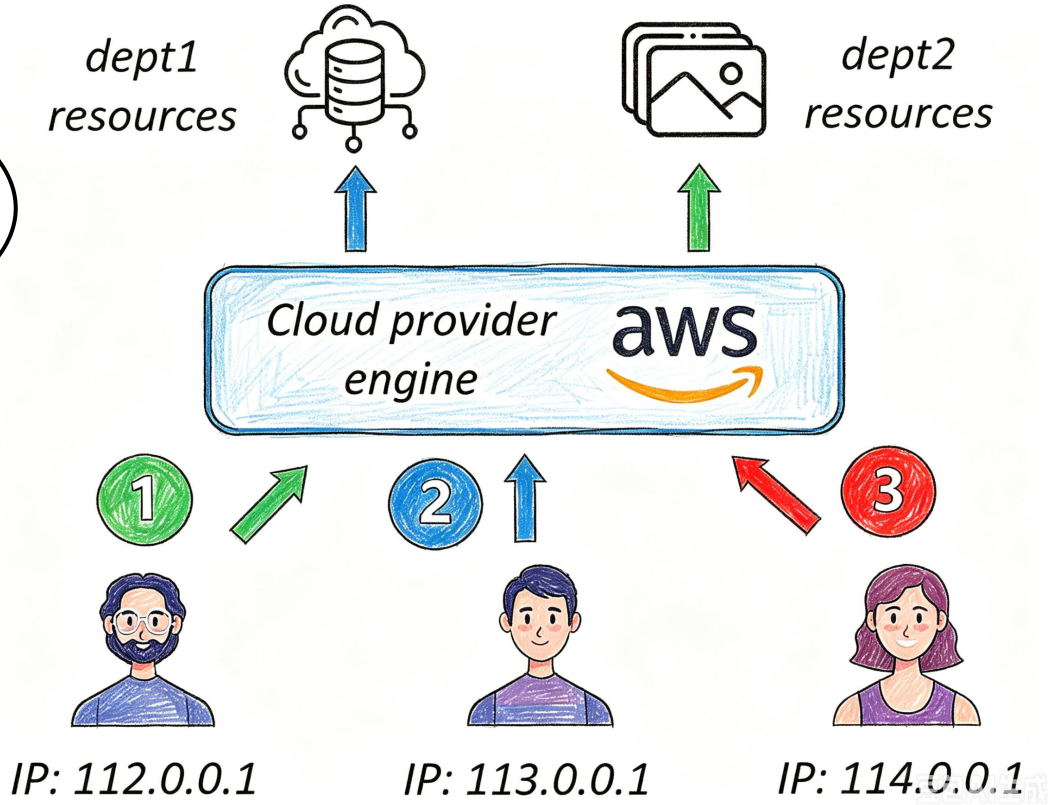
Intent mining can help users

- Access Control Intents
  - *a compact set of declarative statements*

*Describe **who** can perform **which** actions on **what** resources*

Intent1
Resource : *dept*/user1.txt*,
IpAddress : *112.0.0.0/24*

Intent2
Resource : *dept1/user*.txt,*
IpAddress : *113.0.0.0/24*



AWS Access Analyzer has low scalability and readability

- Access Analyzer adopts a stratified **refinement process** based on predefined labels.

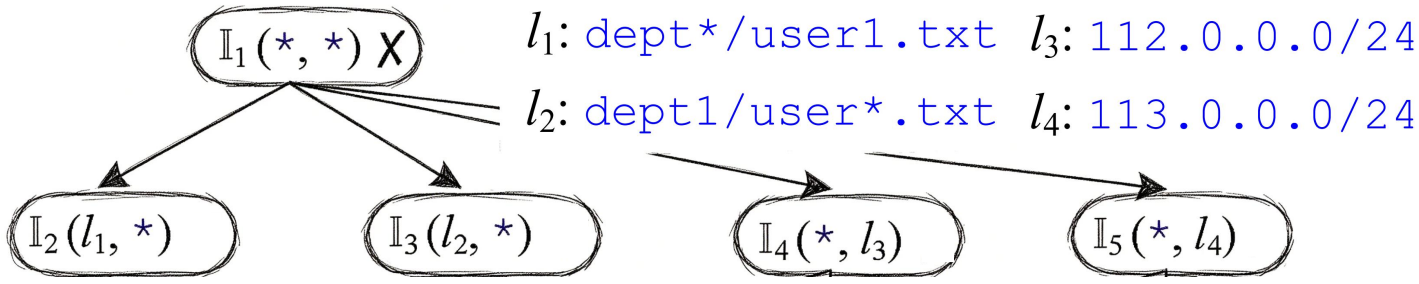
l_1: `dept*/user1.txt` l_3: `112.0.0.0/24`

l_2: `dept1/user*.txt` l_4: `113.0.0.0/24`

AWS Access Analyzer has low scalability and readability

- Access Analyzer adopts a stratified **refinement process** based on predefined labels.
  - *It iteratively checks the following constraint*
  - *If the constraint is unsatisfiable, its child intents are added to the worklist*

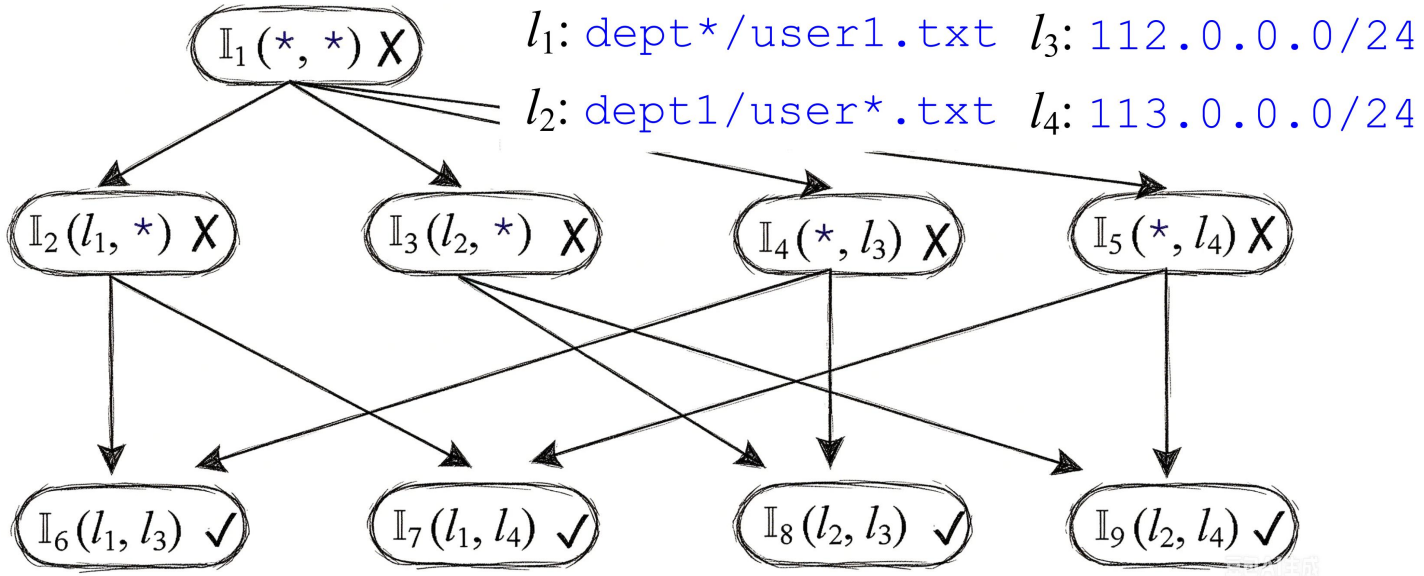
$$\mathbb{P} \wedge \left(\mathbb{I}_i \wedge \neg \left(\bigvee_{\mathbb{I}_j \in \text{Child}(\mathbb{I}_i)} \mathbb{I}_j \right) \right)$$



AWS Access Analyzer has low scalability and readability

- Access Analyzer adopts a stratified **refinement process** based on predefined labels.
  - *It iteratively checks the following constraint*
  - *If the constraint is unsatisfiable, its child intents are added to the worklist*
  - *Otherwise, the intent is included in the final set of intents*

$$\mathbb{P} \wedge \left(\mathbb{I}_i \wedge \neg \left(\bigvee_{\mathbb{I}_j \in \text{Child}(\mathbb{I}_i)} \mathbb{I}_j \right) \right)$$



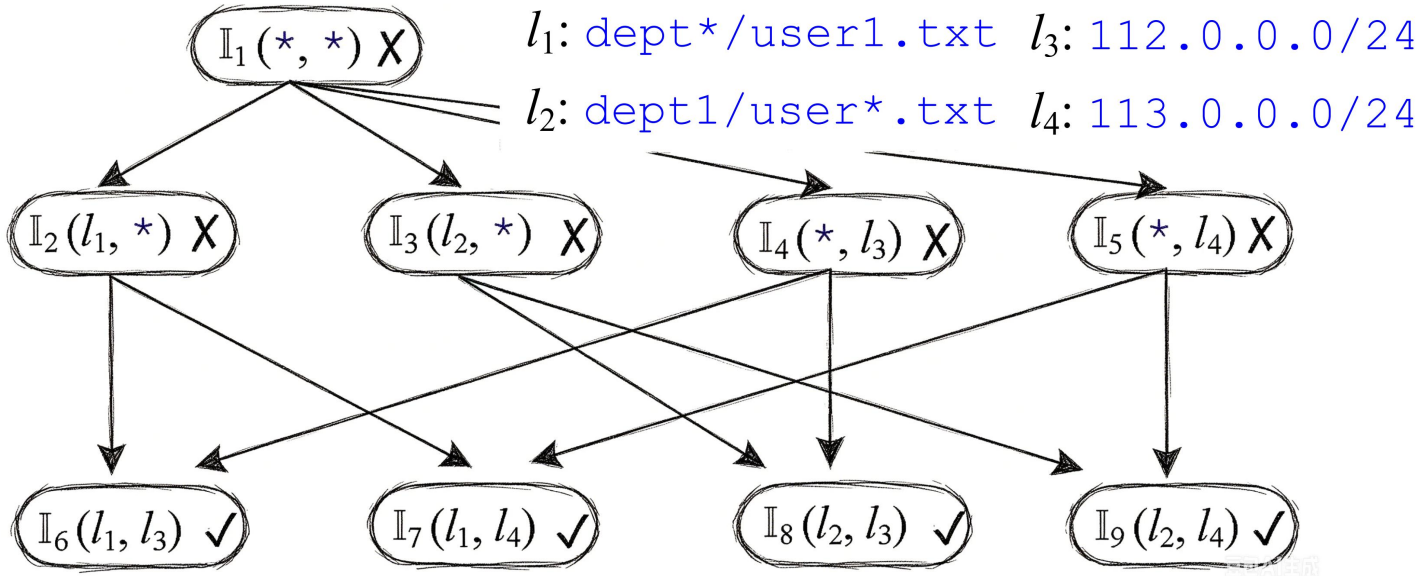
AWS Access Analyzer has low scalability and readability

- Access Analyzer adopts a stratified **refinement process** based on predefined labels.
  - *It iteratively checks the following constraint*
  - *If the constraint is unsatisfiable, its child intents are added to the worklist*
  - *Otherwise, the intent is included in the final set of intents*

$$\mathbb{P} \wedge \left(\mathbb{I}_i \wedge \neg \left(\bigvee_{\mathbb{I}_j \in \text{Child}(\mathbb{I}_i)} \mathbb{I}_j \right) \right)$$

- SMT solving round = 9
- The number of intents is 4

But we only need Intent6 and Intent9. All other intents are covered by them.



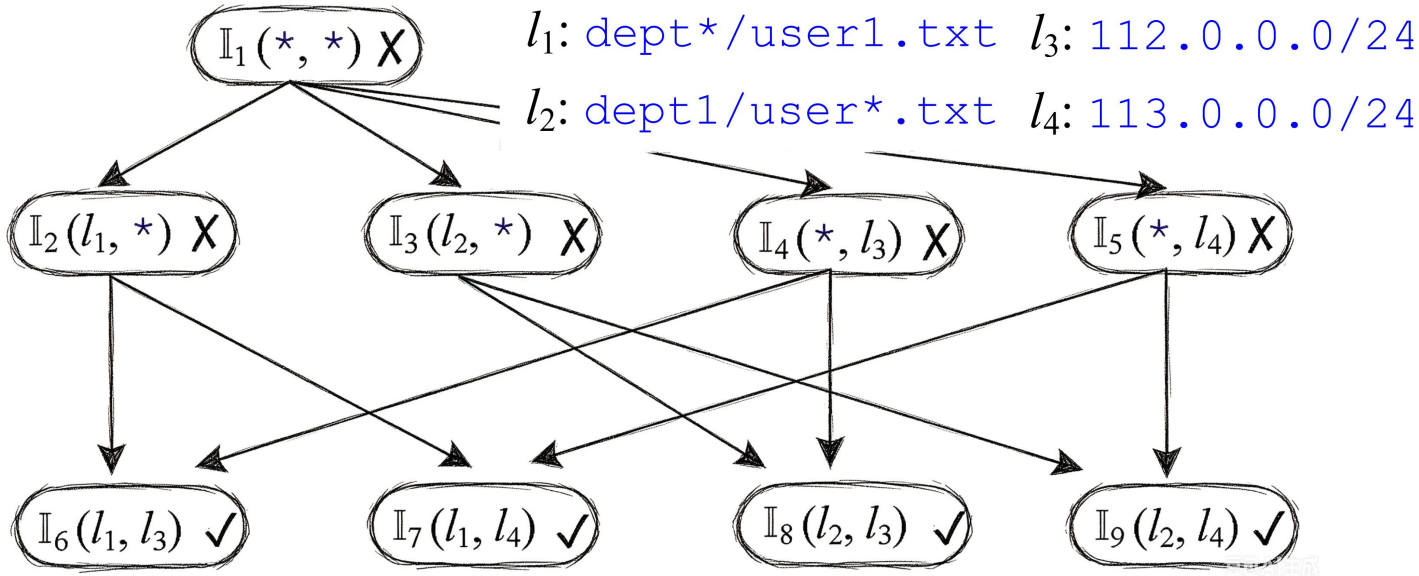
AWS Access Analyzer has low scalability and readability

- Access Analyzer adopts a stratified **refinement process** based on predefined labels.
  - *It iteratively checks the following constraint*
  - *If the constraint is unsatisfiable, its child intents are added to the worklist*
  - *Otherwise, the intent is included in the final set of intents*

$$\mathbb{P} \wedge \left(\mathbb{I}_i \wedge \neg \left(\bigvee_{\mathbb{I}_j \in \text{Child}(\mathbb{I}_i)} \mathbb{I}_j \right) \right)$$

- A policy with 15 statements

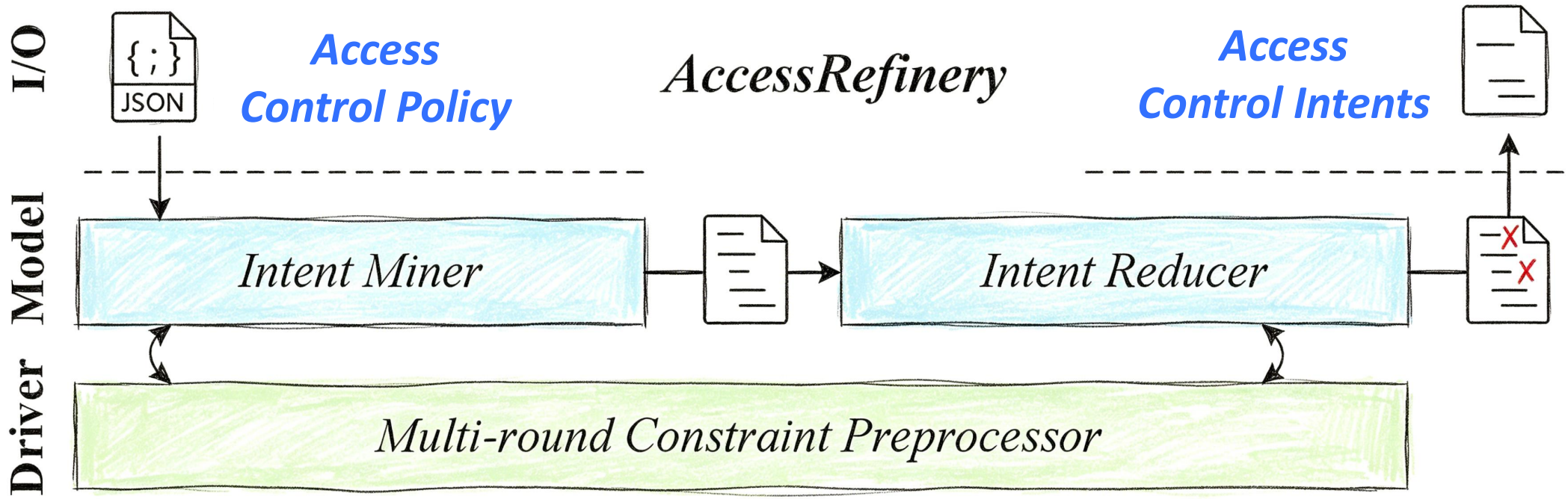
- Mining time exceeds 1 hour
  - SMT solving round > 1000
  - Single SMT solving is complex
- The number of intents reaches 225



Can we mine access control intents within a ***reasonable time*** *while maintaining* ***conciseness****?*

AccessRefinery

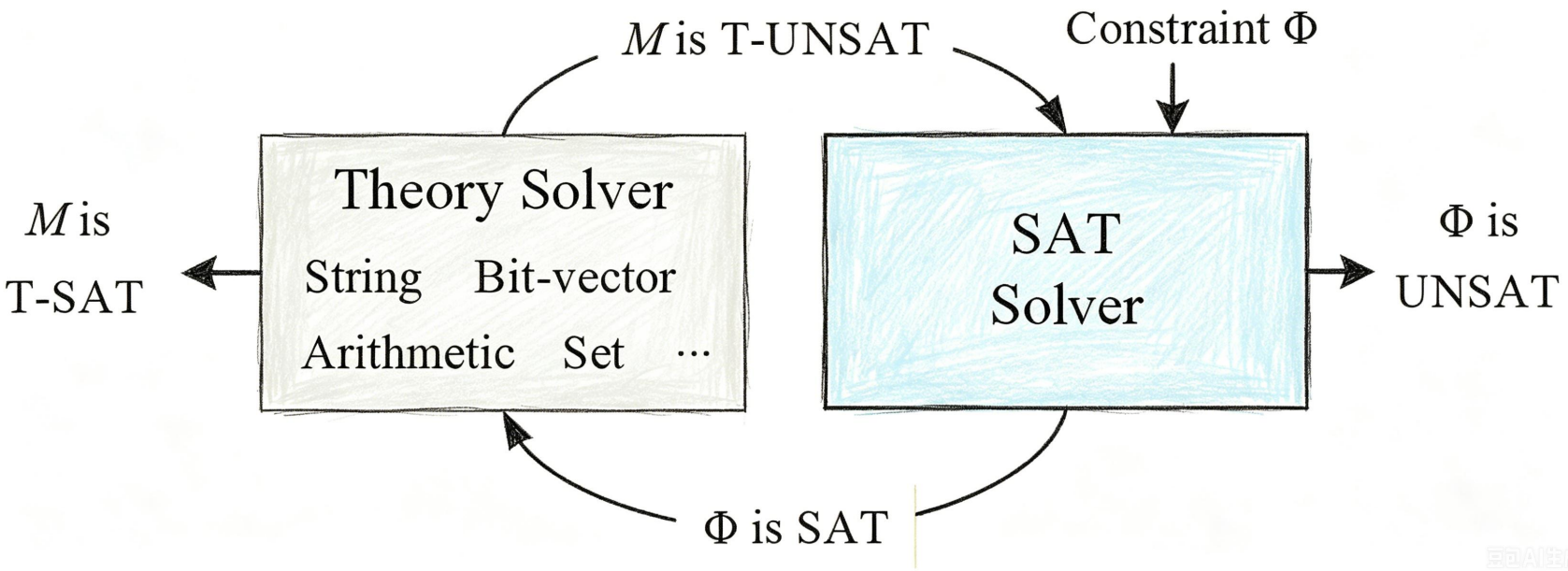
- AccessRefinery takes **Access Control Policy** as input and produces **Access Control Intents**.
  - ***MCP*** *decouples the underlying Boolean operations from the higher-level application logic.*
  - ***MCP*** *allows the upper-layer algorithms to focus solely on their logic.*



Multi-round Constraint Preprocessor

Background on SMT Solvers

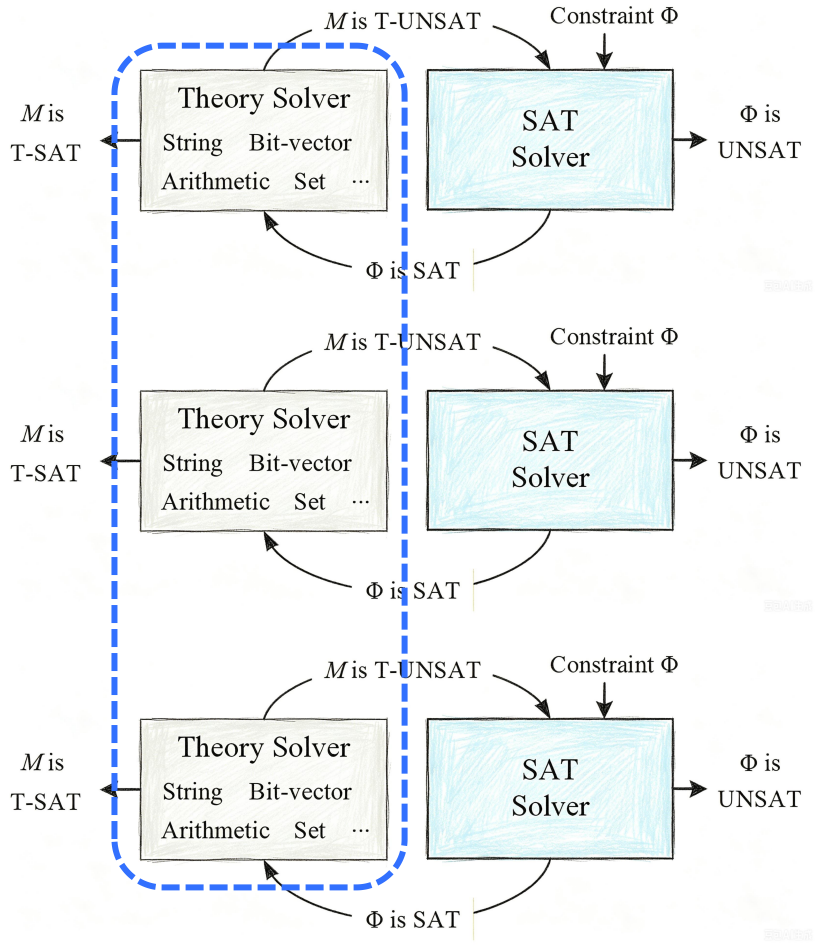
- SMT solving per query is **expensive** due to the iterative process between theory solvers and the SAT solver.



Insight of MCP

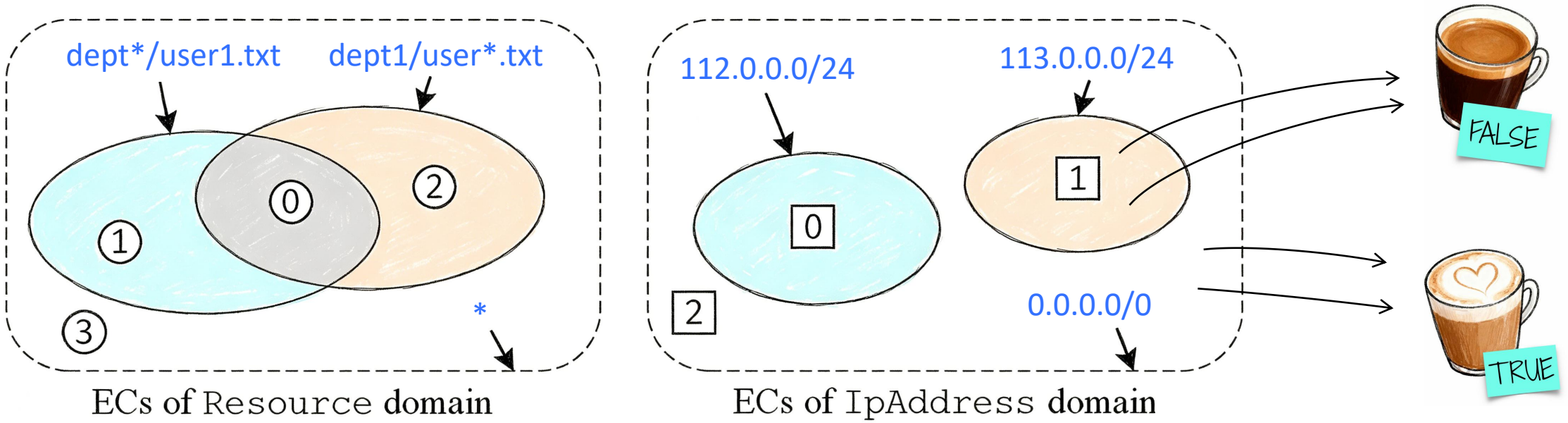
- When performing multi-round SMT solving, theory solving is repeatedly invoked in every round, resulting in substantial redundancy.

Can we reduce this redundancy?



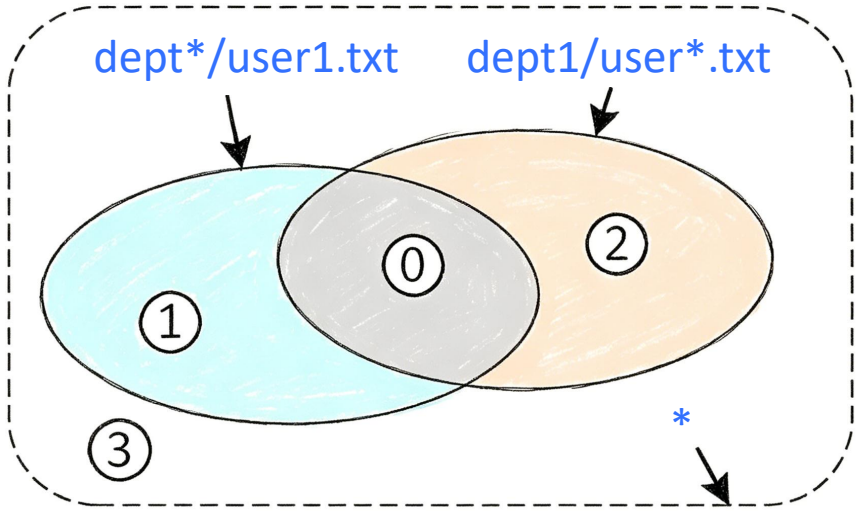
Multi-round Constraint Preprocessor

- Extract Redundancies for theory solvers in advance, thereby avoiding repeated theory solving across multiple SMT solving rounds.
  - *It enables partitioning these possible values (i.e., labels) into disjoint spaces, i.e., equivalence classes (ECs).*
  - *Two values in the same EC yield the same satisfiability result for the theory solver.*

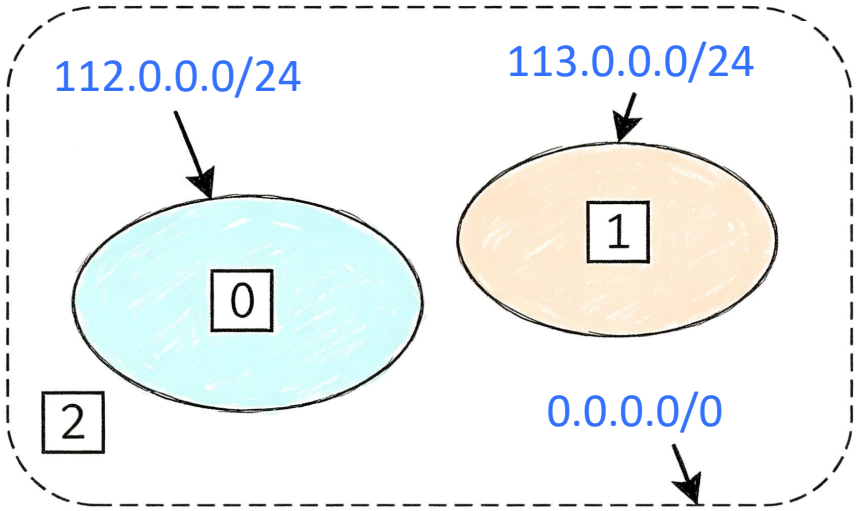


Multi-round Constraint Preprocessor

- Extract Redundancies for theory solvers in advance, thereby avoiding repeated theory solving across multiple SMT solving rounds.
  - *We can map each label to its corresponding ECs before solving.*
  - *Pre-computing of equivalence classes is performed only once, and the results are stored in the mapping.*



ECs of Resource domain



ECs of IpAddress domain

- dept*/user1 ↦ {⓪, ①}
- dept1/user* ↦ {⓪, ②}
- * ↦ {⓪, ①, ②, ③}

Multi-round Constraint Preprocessor

- MCP obtain single-theory constraints over bit-vectors that are equivalent to multi-theory SMT constraints.

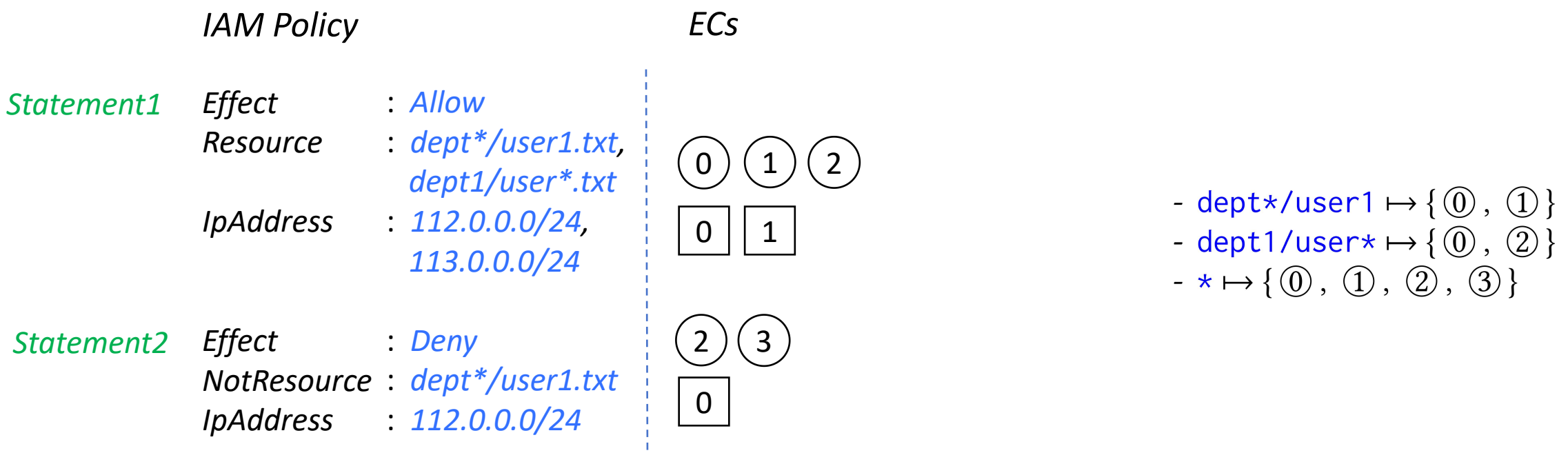
IAM Policy

Statement1	*Effect*	: *Allow*
	Resource	: *dept*/user1.txt*, *dept1/user*.txt*
	IpAddress	: *112.0.0.0/24*, *113.0.0.0/24*
Statement2	*Effect*	: *Deny*
	NotResource	: *dept*/user1.txt*
	IpAddress	: *112.0.0.0/24*

- `dept*/user1` $\mapsto \{⓪, ①\}$
- `dept1/user*` $\mapsto \{⓪, ②\}$
- `*` $\mapsto \{⓪, ①, ②, ③\}$

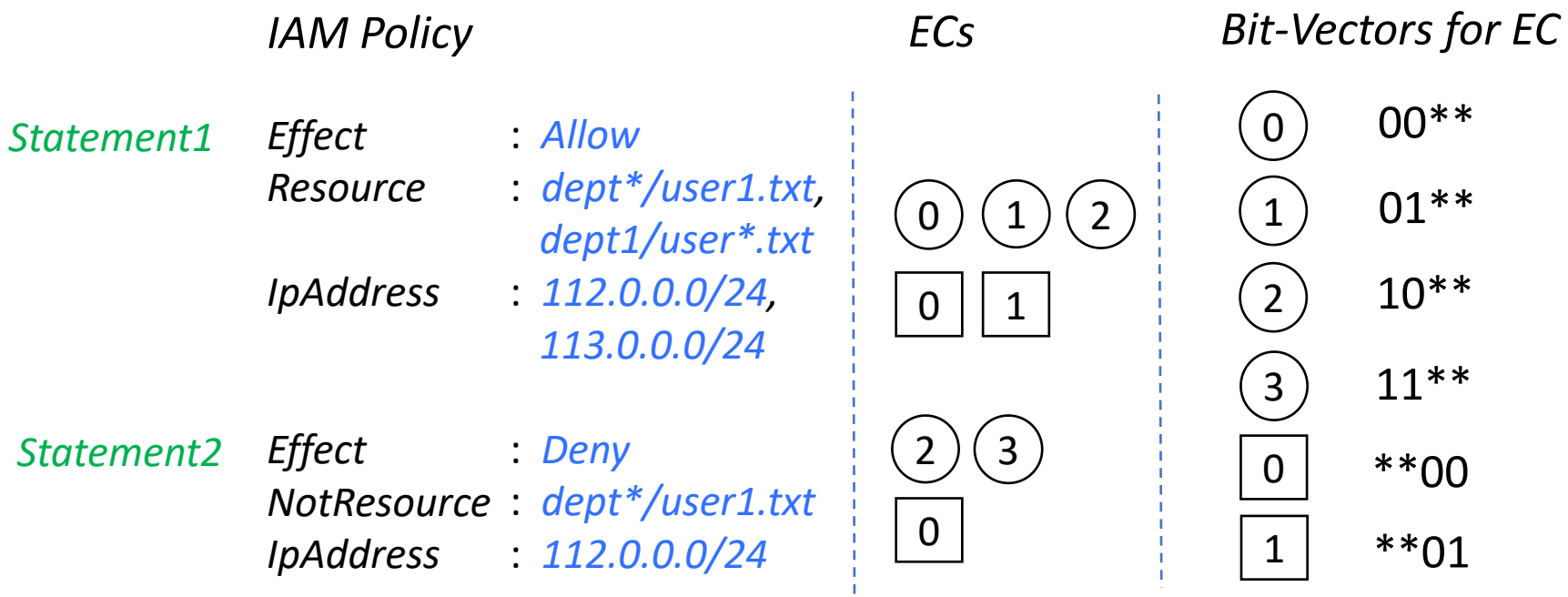
Multi-round Constraint Preprocessor

- MCP obtain single-theory constraints over bit-vectors that are equivalent to multi-theory SMT constraints.



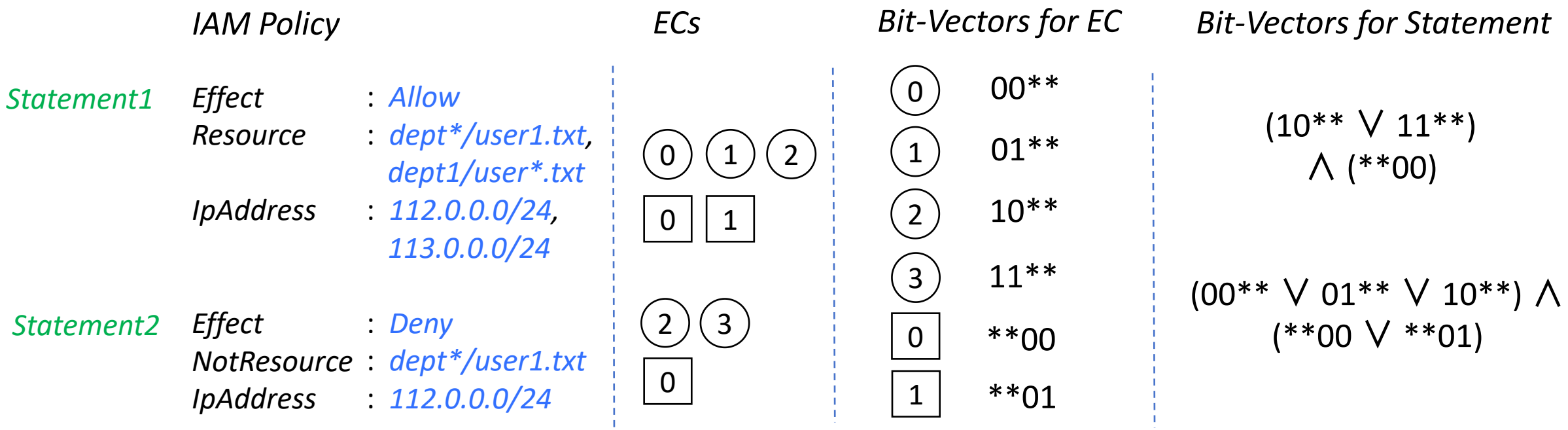
Multi-round Constraint Preprocessor

- MCP obtain single-theory constraints over bit-vectors that are equivalent to multi-theory SMT constraints.



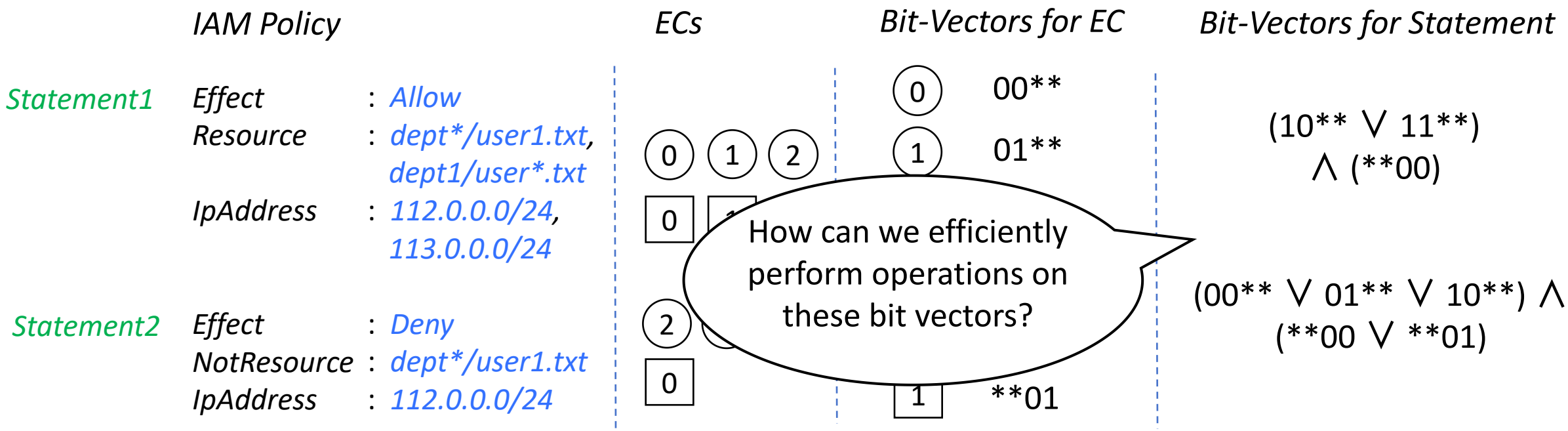
Multi-round Constraint Preprocessor

- MCP obtain single-theory constraints over bit-vectors that are equivalent to multi-theory SMT constraints.



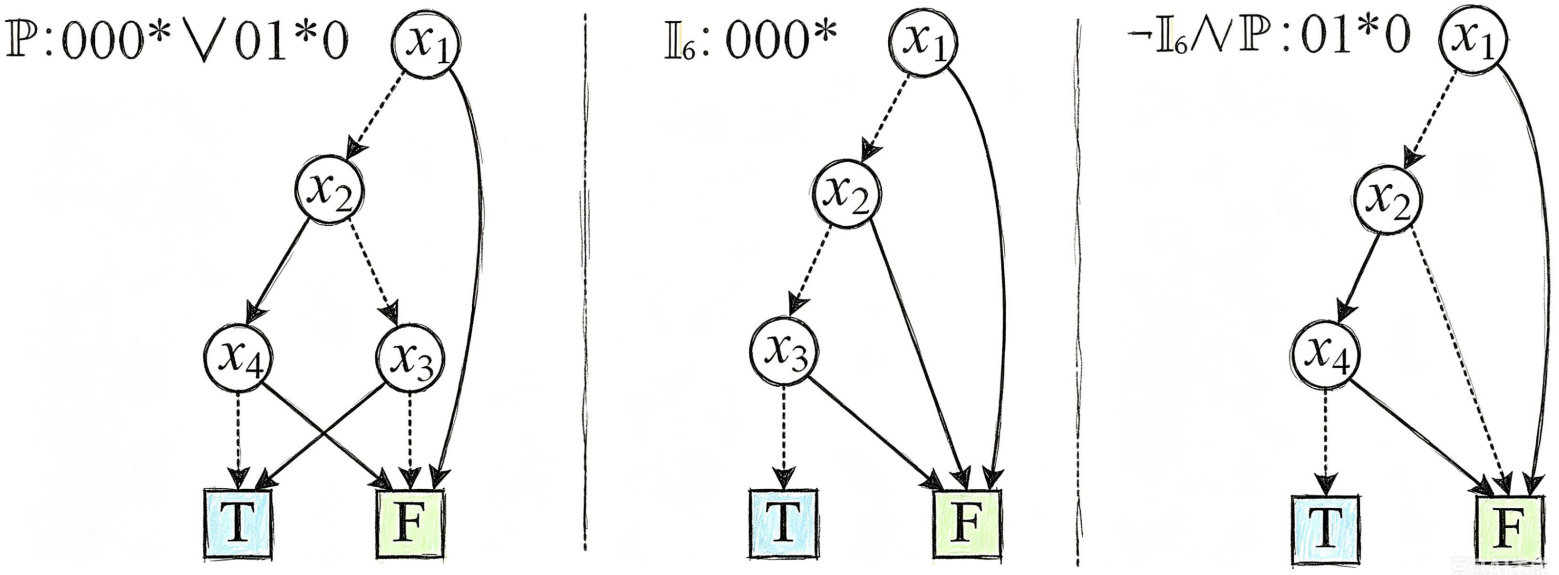
Multi-round Constraint Preprocessor

- MCP obtain single-theory constraints over bit-vectors that are equivalent to multi-theory SMT constraints.



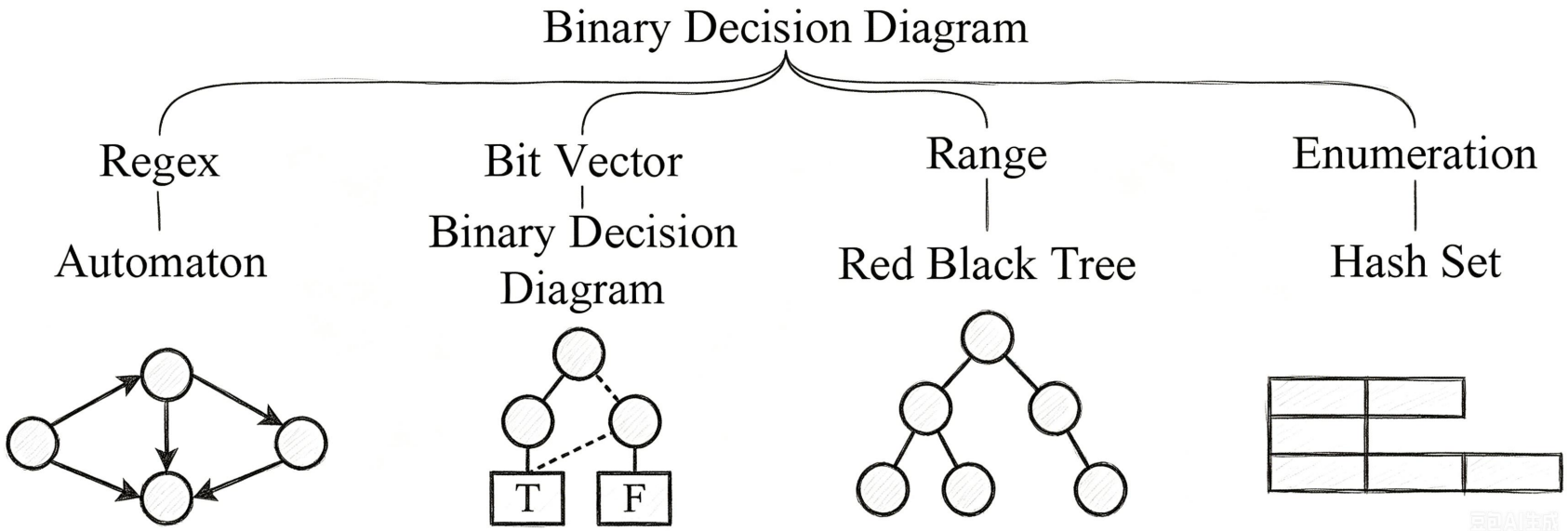
Multi-round Constraint Preprocessor

- We adopt Binary Decision Diagrams (BDDs) to compactly represent bit-vectors instead of using the SAT solver.
  - *BDD is a directed acyclic graph (DAG) with terminal nodes (true and false) and variable nodes.*
  - *The insight is that BDDs allow more efficient re-use of intermediate results for* ***incremental computation****.*



The usage of MCP

- We design MCP as a data structure that we believe will also benefit other studies MCP supports different types of value.
  - *MCP hides the underlying details and supports fast multi-round SMT solving.*
  - *MCP supports multiple type of values, including **regex, bit-vector, range and enumeration**.*



Open-sourced at https://github.com/XJTU-NetVerify/accessrefinery/

The usage of MCP

- An example using MCP to solve :

  $\neg Intent6 \wedge Policy$

The initials of MCP

```
/* Declare the domain values. */
MCP mcp = new MCPFactory();
mcp.addValue("Res", REGEXP, "dept*/user1.txt");
mcp.addValue("Res", REGEXP, "dept1/user*.txt");
mcp.addValue("IP",  PREFIX, "112.0.0.0/24");
mcp.addValue("IP",  PREFIX, "113.0.0.0/24");
/* Extract theory redundancy.*/
mcp.partitionECs();
```

The usage of MCP

- An example using MCP to solve :

¬Intent6 ∧ Policy

The initials of MCP

Multi-round SMT solving

Statements

```
/* Declare the domain values. */
MCP mcp = new MCPFactory();
mcp.addValue("Res", REGEXP, "dept*/user1.txt");
mcp.addValue("Res", REGEXP, "dept1/user*.txt");
mcp.addValue("IP",  PREFIX, "112.0.0.0/24");
mcp.addValue("IP",  PREFIX, "113.0.0.0/24");
/* Extract theory redundancy.*/
mcp.partitionECs();
/* Multi-round SMT solving. */
BDD res1  = mcp.getValue("Res", "dept*/user1.txt");
BDD res2  = mcp.getValue("Res", "dept1/user*.txt");
BDD ip1 = mcp.getValue("IP", "112.0.0.0/24");
BDD ip2 = mcp.getValue("IP", "113.0.0.0/24");
BDD s1 = (res1.or(res2)).and(ip1.or(ip2));
BDD s2 = res1.not().and(ip1);
BDD s3 = res2.not().and(ip2);
```

The usage of MCP

- An example using MCP to solve : ***¬Intent6 ⋀ Policy***

The initials of MCP

Multi-round SMT solving

Statements

Policy

¬Intent6

¬Intent6 ⋀ Policy

```
/* Declare the domain values. */
MCP mcp = new MCPFactory();
mcp.addValue("Res", REGEXP, "dept*/user1.txt");
mcp.addValue("Res", REGEXP, "dept1/user*.txt");
mcp.addValue("IP",  PREFIX, "112.0.0.0/24");
mcp.addValue("IP",  PREFIX, "113.0.0.0/24");
/* Extract theory redundancy.*/
mcp.partitionECs();
/* Multi-round SMT solving. */
BDD res1  = mcp.getValue("Res", "dept*/user1.txt");
BDD res2  = mcp.getValue("Res", "dept1/user*.txt");
BDD ip1 = mcp.getValue("IP", "112.0.0.0/24");
BDD ip2 = mcp.getValue("IP", "113.0.0.0/24");
BDD s1 = (res1.or(res2)).and(ip1.or(ip2));
BDD s2 = res1.not().and(ip1);
BDD s3 = res2.not().and(ip2);
BDD policy = s1.diff(s2).diff(s3);
BDD intent6 = res1.and(ip1);
if (policy.and(intent6.not()).sat() == true) {...}
```

Intent Reducer

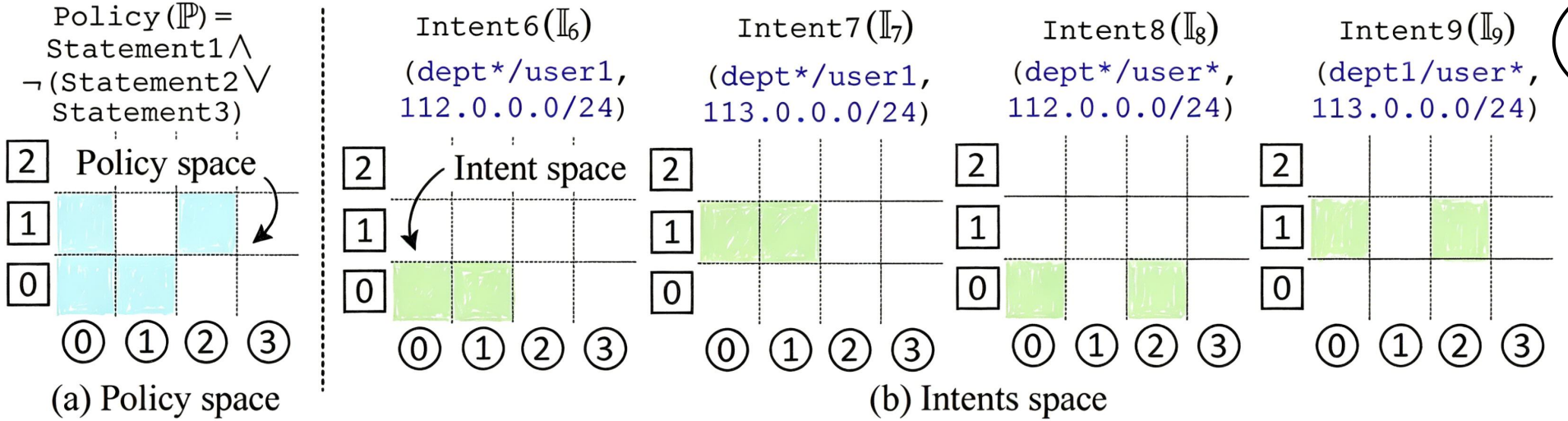
Intent Reducer

- *Reducing intents requires covering all allowed requests of the policy*
  - *Reducing intents can then be formulated as a **min-set-cover problem**.*
  - *However, both policy and intents can potentially **span infinite spaces**.*

Policy($\mathbb{P}$) = Statement1 $\wedge$ $\neg$(Statement2 $\vee$ Statement3)	Intent6($\mathbb{I}_6$) (dept*/user1, 112.0.0.0/24)	Intent7($\mathbb{I}_7$) (dept*/user1, 113.0.0.0/24)	Intent8($\mathbb{I}_8$) (dept*/user*, 112.0.0.0/24)	Intent9($\mathbb{I}_9$) (dept1/user*, 113.0.0.0/24)

Intent Reducer

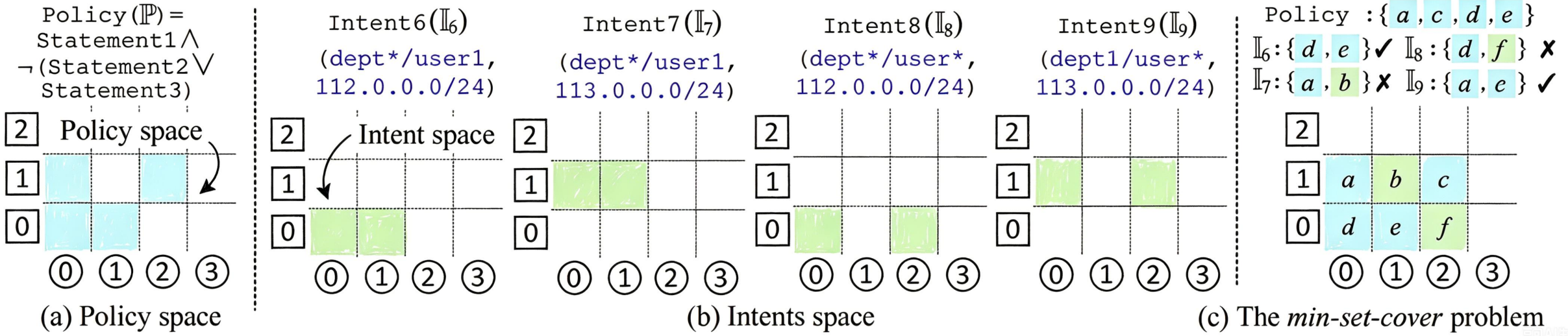
- *Reducing intents requires covering all allowed requests of the policy*
  - *Reducing intents can then be formulated as a **min-set-cover problem**.*
  - *However, both policy and intents can potentially **span infinite spaces**.*
  - *We employ **equivalence class (EC)** techniques to transform the min-set-cover problem over infinite sets into an equivalent problem **over finite set**.*



MCP can efficiently finish this task.

Intent Reducer

- *Reducing intents requires covering all allowed requests of the policy*
  - *Reducing intents can then be formulated as a **min-set-cover problem**.*
  - *However, both policy and intents can potentially **span infinite spaces**.*
  - *We employ **equivalence class (EC)** techniques to transform the min-set-cover problem over infinite sets into an equivalent problem **over finite set**.*



Intent Reducer

- *Reducing intents requires covering all allowed requests of the policy*
  - *Reducing intents can then be formulated as a **min-set-cover problem**.*
  - *However, both policy and intents can potentially **span infinite spaces**.*
  - *We employ **equivalence class (EC)** techniques to transform the min-set-cover problem over infinite sets into an equivalent problem **over finite set**.*
  - *Since each policy and intent maps to only a small number of ECs, we can solve the **min-set-cover problem** problem **exactly using ILP solver**.*

$$\text{minimize} \quad \sum_{i=1}^{n} x_i$$

$$\text{subject to} \quad \left(\sum_{i : c_j \in \mathcal{M}(\mathbb{I}_i)} x_i \right) \geq 1, \quad \forall c_j \in \mathcal{M}(\mathbb{P})$$

$$x_i \in \{0, 1\}, \quad \forall i = 1, \ldots, n$$

Datasets

- *Real world dataset **506 policies***
  - *collected over one year from a customer of **a large cloud provider***
- *Synthetic datasets 1 for the **correctness** experiment*
  - *according to real world policy*
- *Synthetic datasets 2 for the **scalability** experiment*
  - *covering all possible boolean relationships between values*

Implementation

- *AccessRefinery* ***~5k*** *lines of Java code*
- *Access Analyzer* ***~5k*** *lines of Java code (Baseline)*

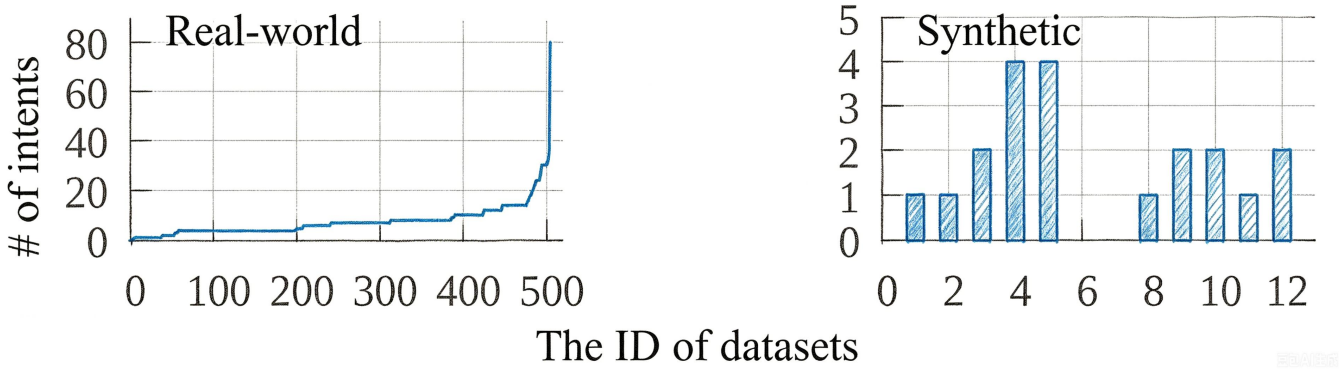
Open-sourced at https://github.com/XJTU-NetVerify/accessrefinery/

Correctness

- Correctness of MCP
  - *we conducted a series of **basic Boolean operations tests***

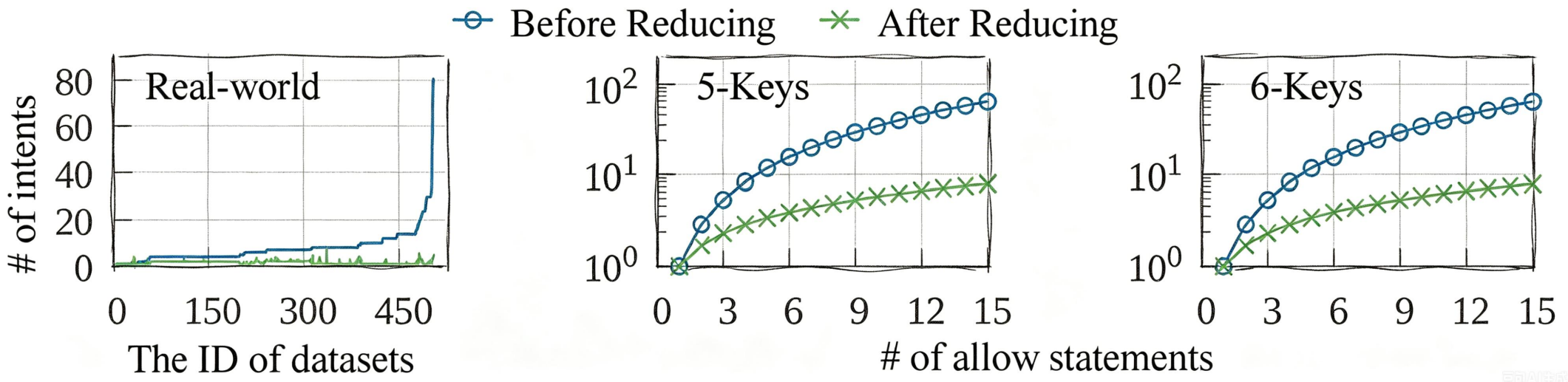
(1) Nested expressions: $(A \wedge B) \vee (\neg A \wedge C)$;
(2) Implication chains: $(A \rightarrow B) \wedge (B \rightarrow C)$;
(3) Distributive law: $A \wedge (B \vee C)$ vs $(A \wedge B) \vee (A \wedge C)$;
(4) De Morgan's laws: $\neg(A \wedge B)$ vs $\neg A \vee \neg B$;
(5) Tautology and contradiction: $A \vee \neg A$ (tautology) and $A \wedge \neg A$ (contradiction); ...

- Correctness *of Intent Miner*
  - *We compared the intents produced by **AccessRefinery**, our **reproduced Access Analyzer**, and the **AWS Access Analyzer via the CLI API**.*



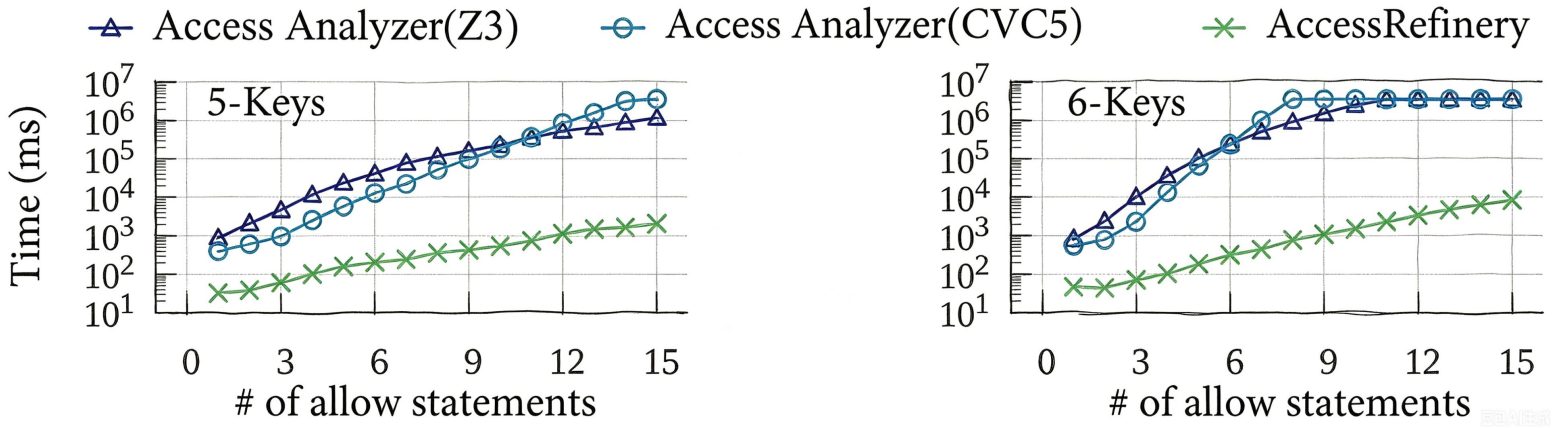
Readability

- Reduce the number of intents by up to ***one order of magnitude*** for both real and synthetic datasets.
  - *the reduction becomes more significant as the number of Allow statements increases.*

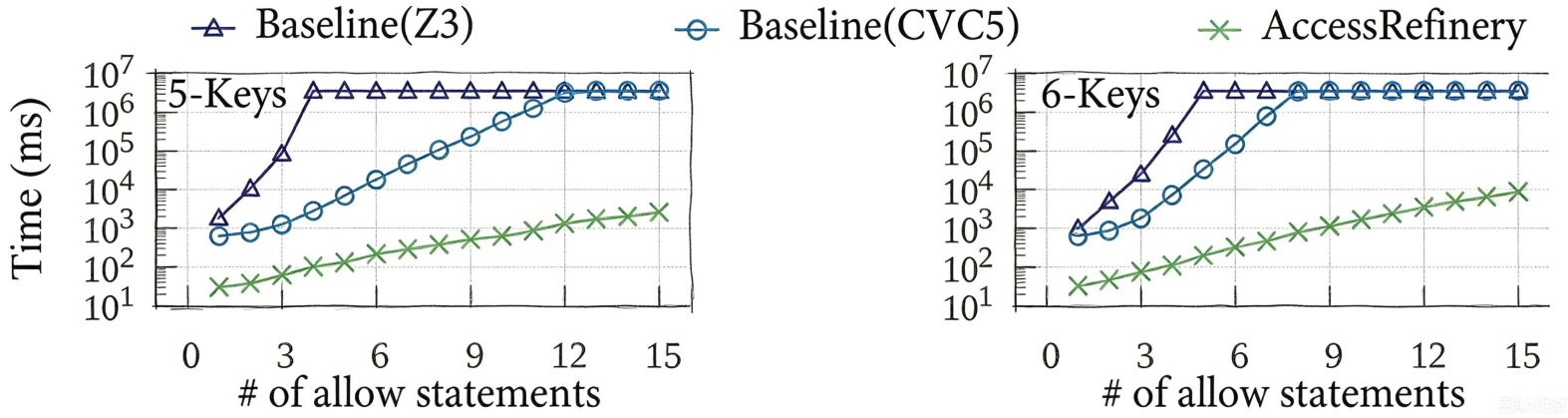


Scalability - Synthetic Datasets

- For intent mining, ***AccessRefinery*** achieves a speedup of ***one to two orders of magnitude***.

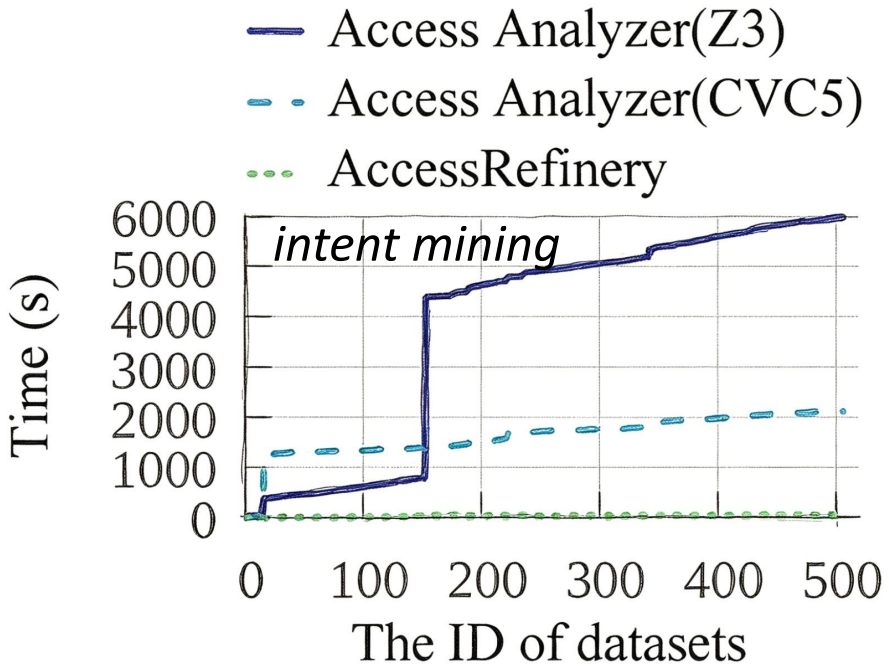


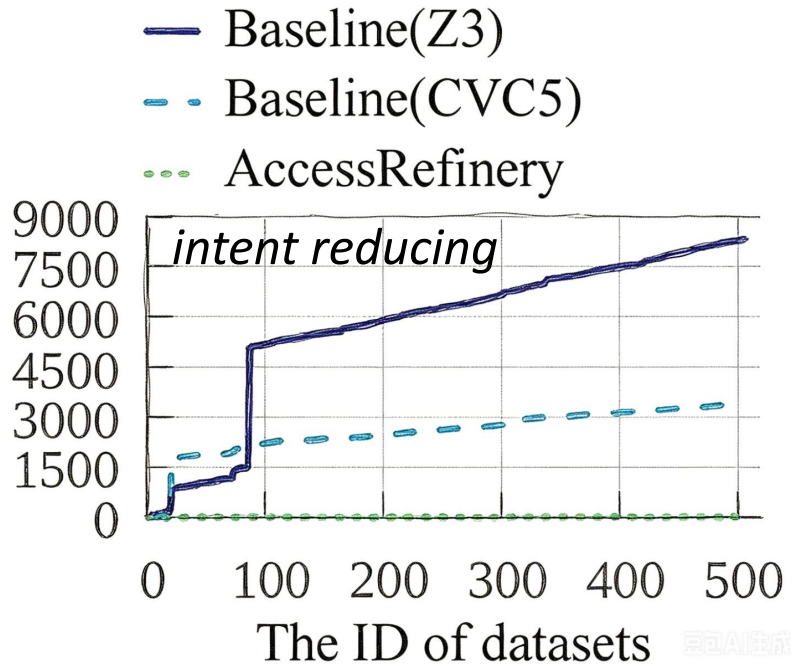
- For intent reduction, ***AccessRefinery*** achieves a speedup of ***one to four orders of magnitude***.



Scalability - Real-world Dataset

- **AccessRefinery** achieves up to **15 × *faster in the intent mining*** and up to **170 × *faster in the intent reduction***.





Microbenchmark

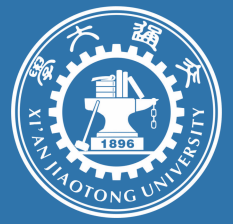
- Through preprocessing SMT constraints, AccessRefinery reduces the time of single-round SMT solving ***from seconds to milliseconds***.

# of allow statements	# of rounds	The time of single-round Boolean solving			The time of MCP preprocessing
		Z3	CVC5	MCP	
3	64	192.1 ms	27.9 ms	77.5 μs	54.5 ms
6	196	756.2 ms	254.3 ms	64.4 μs	189.7 ms
9	400	1517.2 ms	987.0 ms	104.1 μs	394.3 ms
12	676	3122.2 ms	4979.3 ms	190.2 μs	1011.7 ms
15	1024	4545.8 ms	timeout	274.5 μs	1741.5 ms

Conclusion

- Access Control Intents are ***essential*** in Cloud
- Mining access control intents suffers from ***low scalability*** and ***readability***.
- We propose ***AccessRefinery*** to improve scalability and readability
  - *It preprocessing the constraints into bit-vector constraints.*
  - *It computes a compact set of intents that can cover the mined intents.*

- AccessRefinery achieves a **~10–100× *speedup in intent mining***, and reduces the number of intents by up to **~10×**.



西安交通大學
XI'AN JIAOTONG UNIVERSITY

HUAWEI

Happy to take your questions!

The ACM International Conference on the Foundations of Software Engineering (FSE)

Montreal, Canada
Sun 5 - Thu 9 July 2026