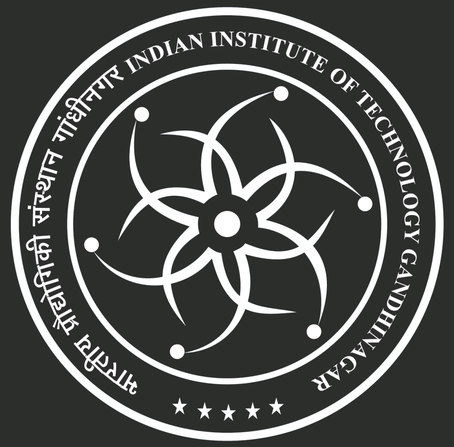


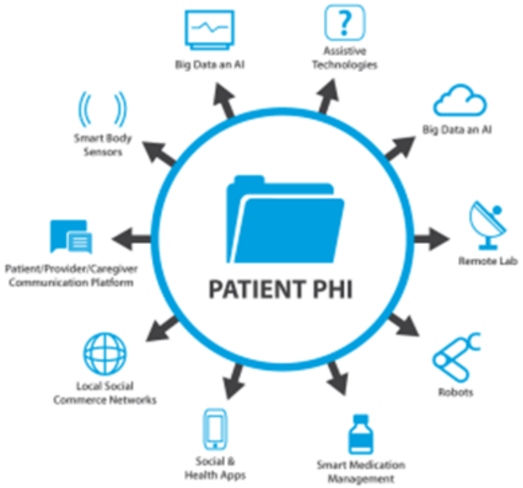
Automating Audit with Policy Inference

Abhishek Bichhawat (IIT Gandhinagar),
Matt Fredrikson (CMU),
and Jean Yang (Akita Software)

Organizational Data Security

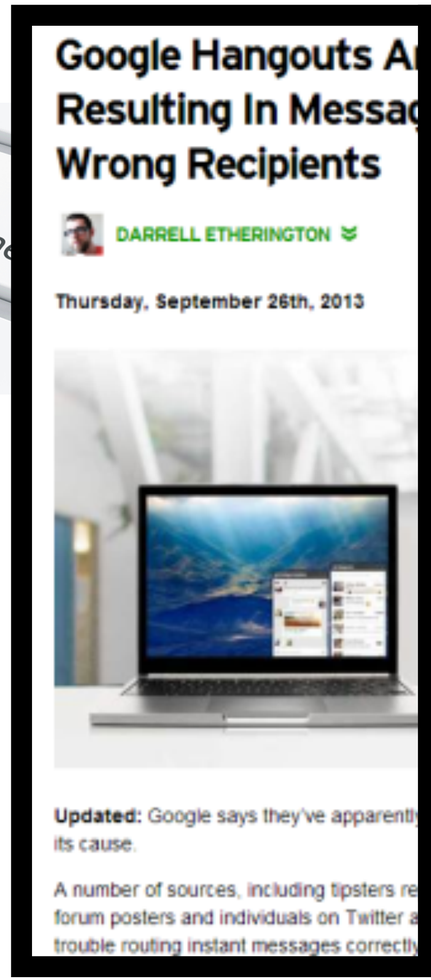
=

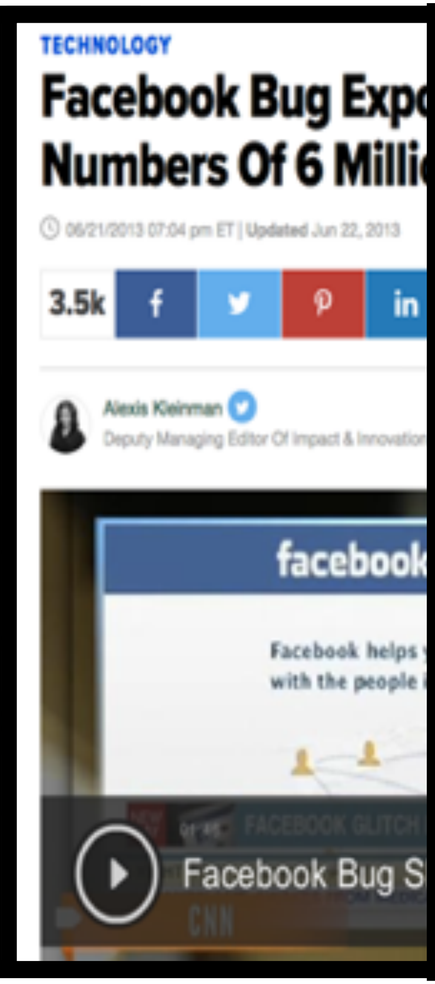
=



=

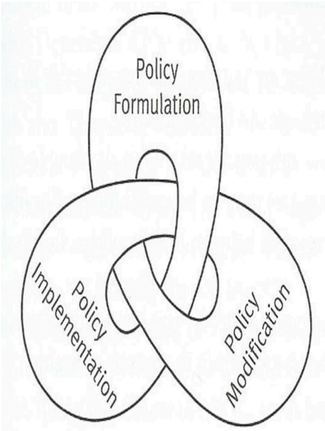
Organizational Data Security



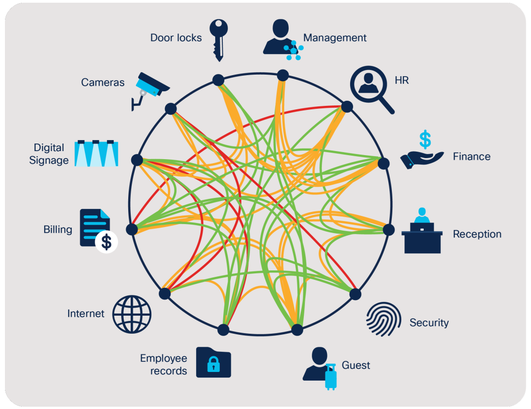


Factors Affecting Policy Enforcement

Constant policy modification

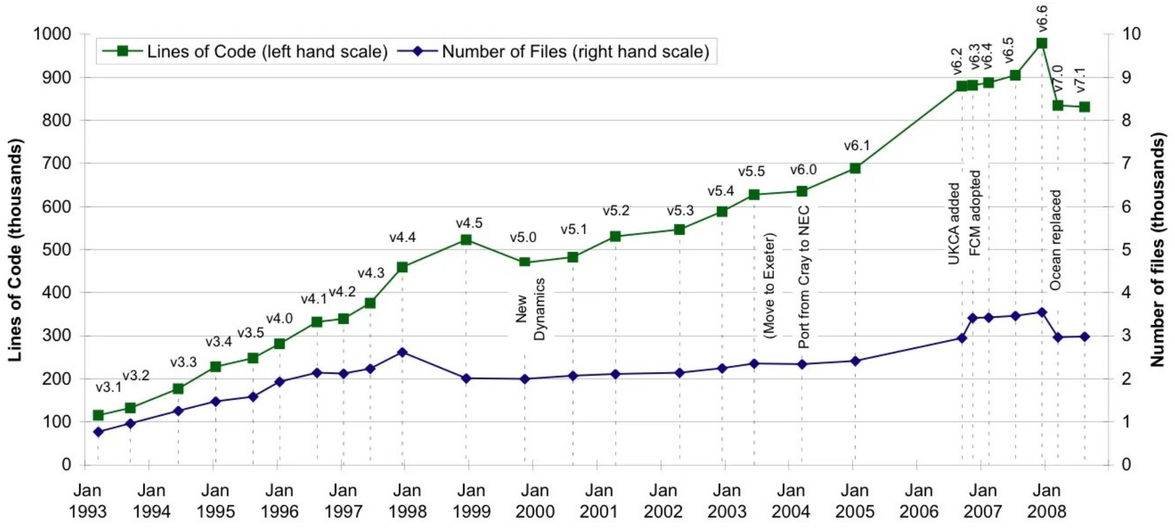


Complexity of policy



Size of the codebase

Evolution of code



Factors Affecting Policy Enforcement

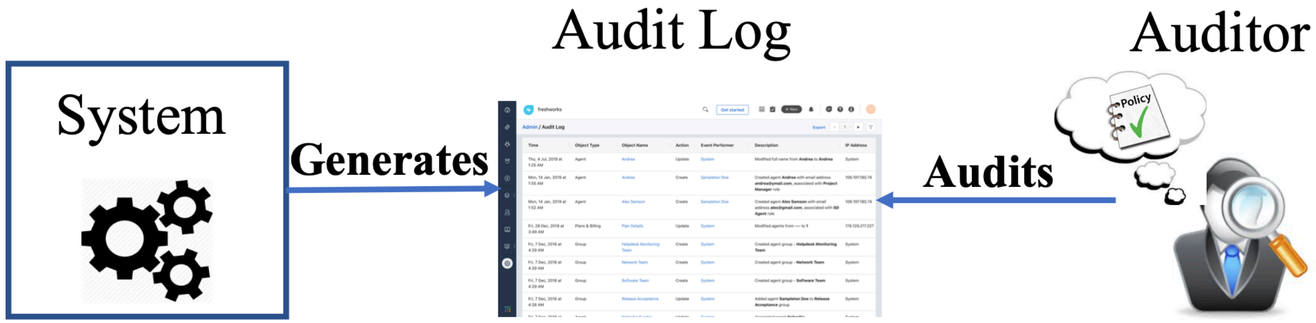
- Organizations may lack explicit, mechanized policy specification

- Inline checks are error-prone; introduce bugs

```
if (in_array($limit, ["a", "r", "ar", "rout", "vis"], true)
    || ($user->privChair && in_array($limit, ["all", "unsub", "unm"], true))
    || ($user->isPC && in_array($limit, ["acc", "reqrevs", "req", "lead", "rable",
                                         "editpref", "manager", "und"], true)))
    /* ok */;
else if ($user->privChair && !$limit && $this->conf->timeUpdatePaper())
    $limit = "all";
else if (($user->privChair && $limit === "act")
         || ($user->isPC
             && in_array($limit, ["", "act", "all", "unm"], true)
             && $this->conf->can_pc_see_active_submissions()))
    $limit = "act";
else if ($user->isPC && in_array($limit, ["", "s", "unm"], true))
    $limit = "s";
else if ($limit === "rable")
    $limit = "r";
else if (!$user->is_reviewer())
    $limit = "a";
else if (!$user->is_author())
    $limit = "r";
else
    $limit = "ar";
```

Auditing Logs for Privacy Compliance

- Organizations audit transactions
  - Log transactions
  - After-the-fact validation of logs
  - Employ human auditor to identify violations of organizational policy

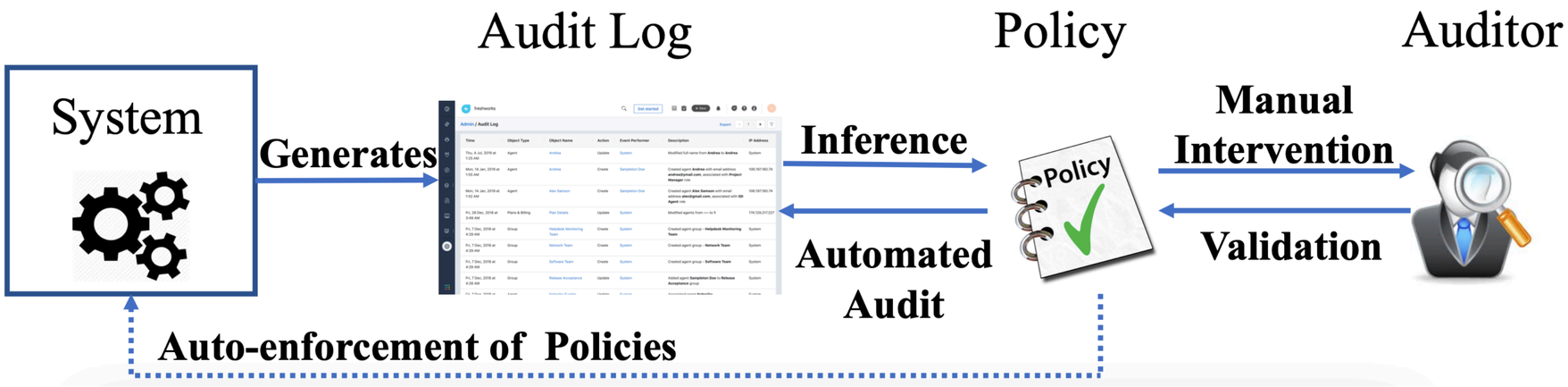


Our Approach - Insight and Overview

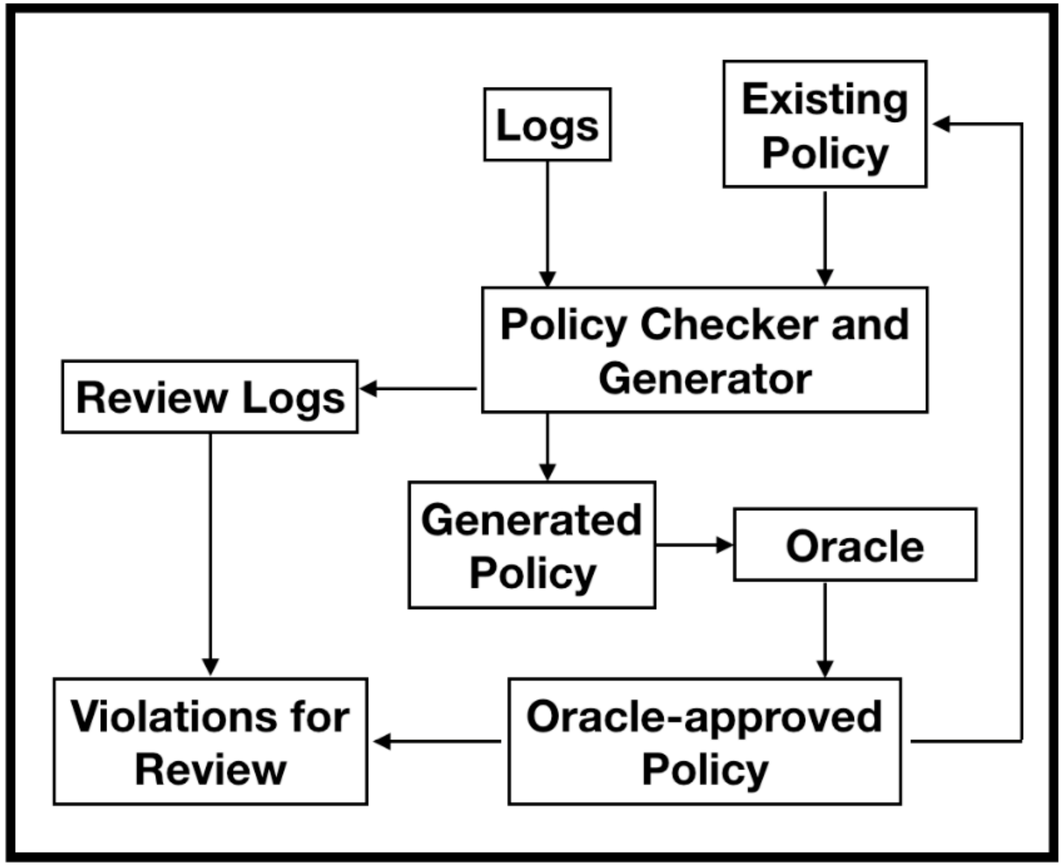
Audit logs combined with auditor-decisions, are sufficient to reconstruct a correct mechanized policy, which can be used to reduce the cost of manually auditing logs

Our Approach - Insight and Overview

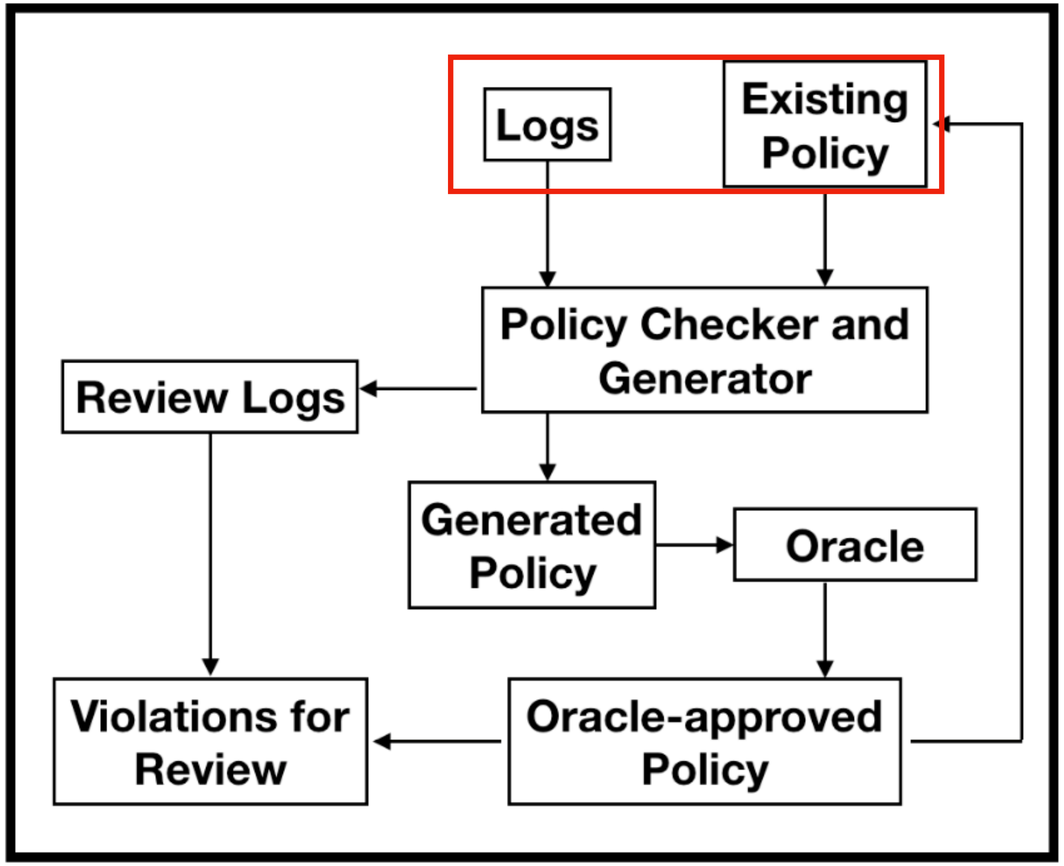
Audit logs combined with auditor-decisions, are sufficient to reconstruct a correct mechanized policy, which can be used to reduce the cost of manually auditing logs



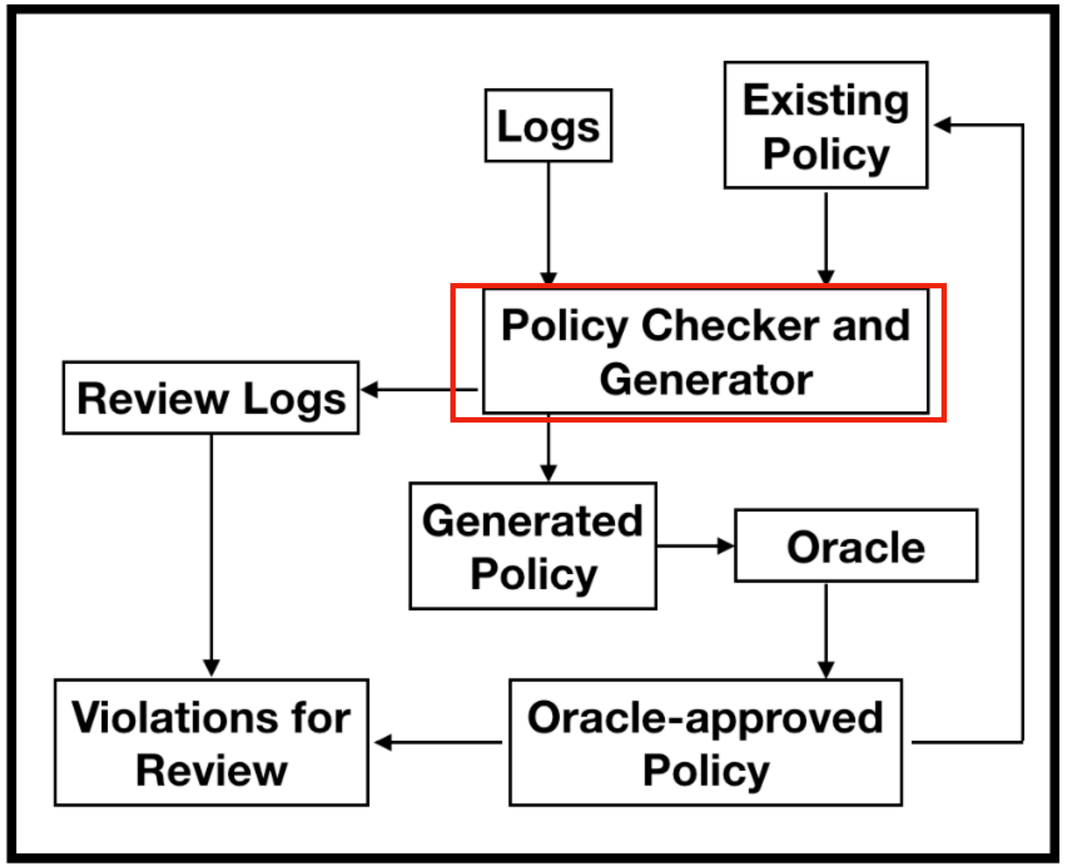
Policy Inference and Audit Framework



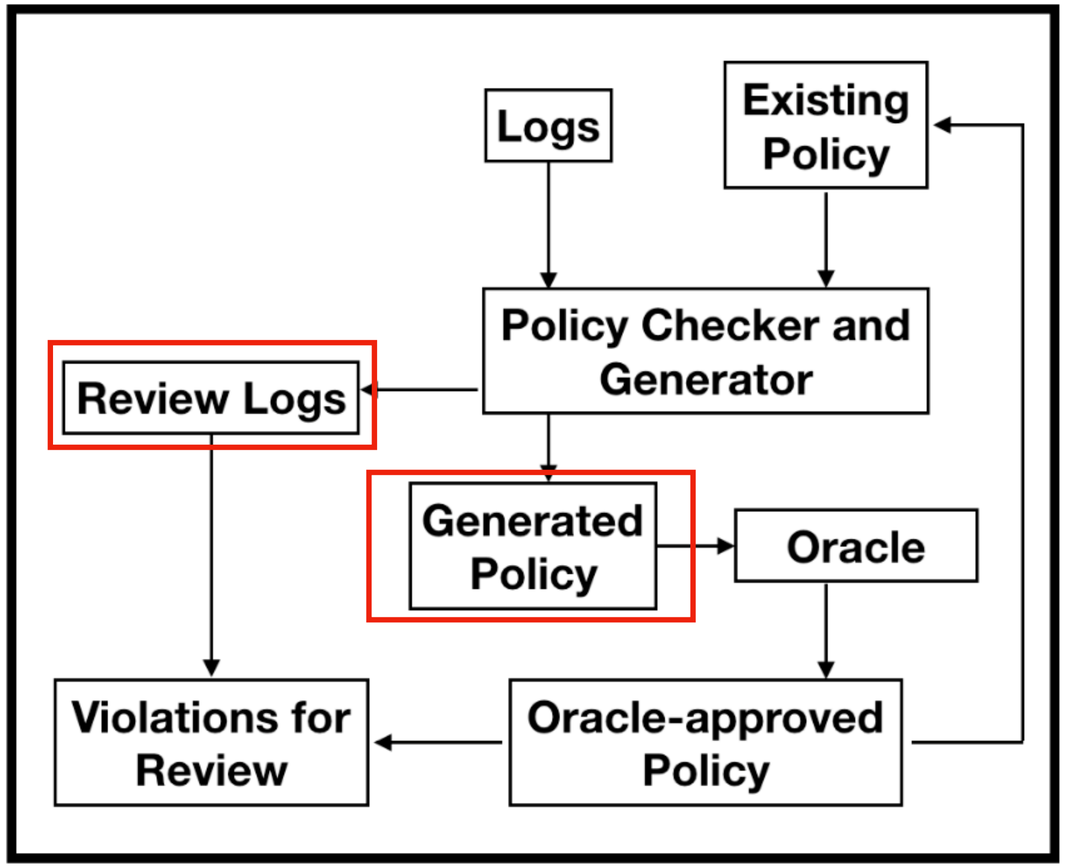
Policy Inference and Audit Framework



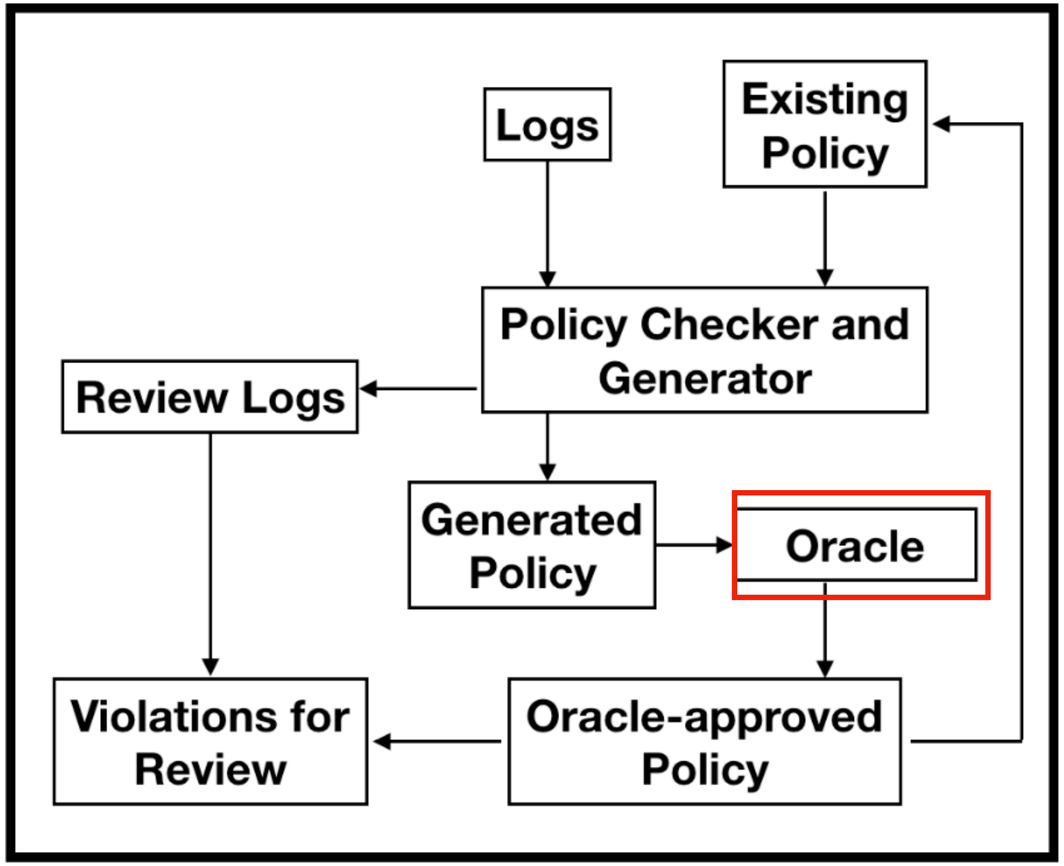
Policy Inference and Audit Framework



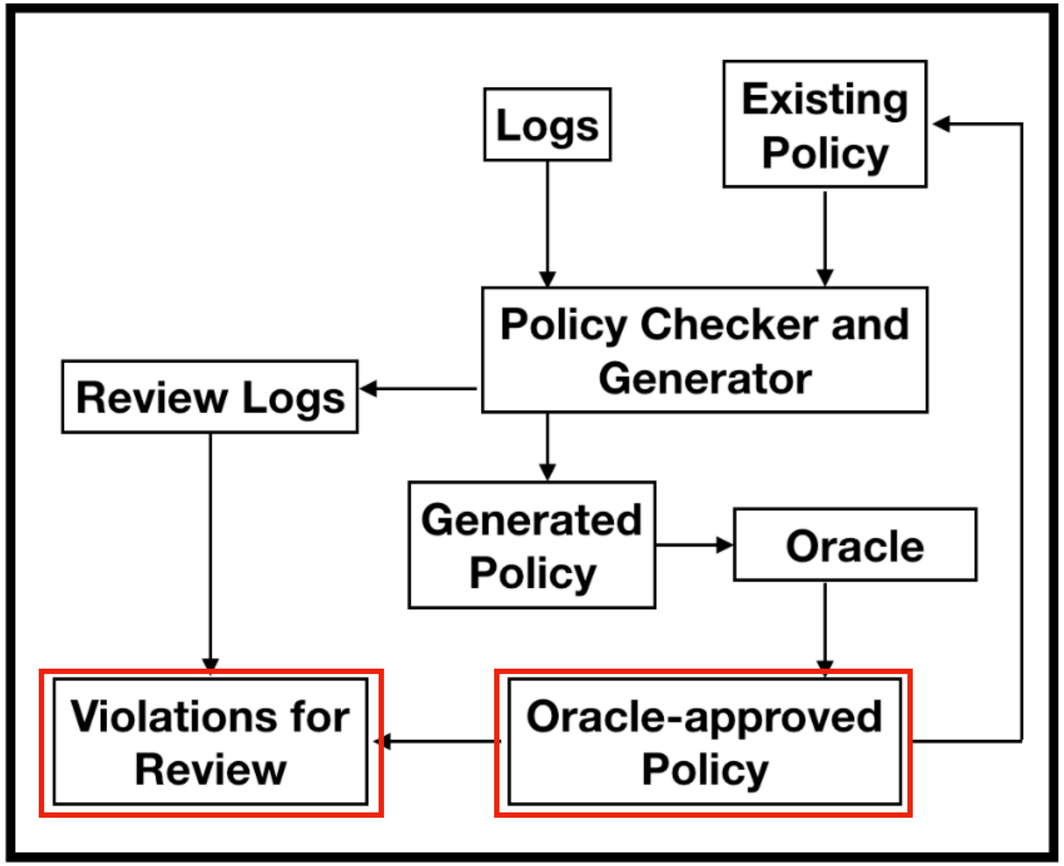
Policy Inference and Audit Framework



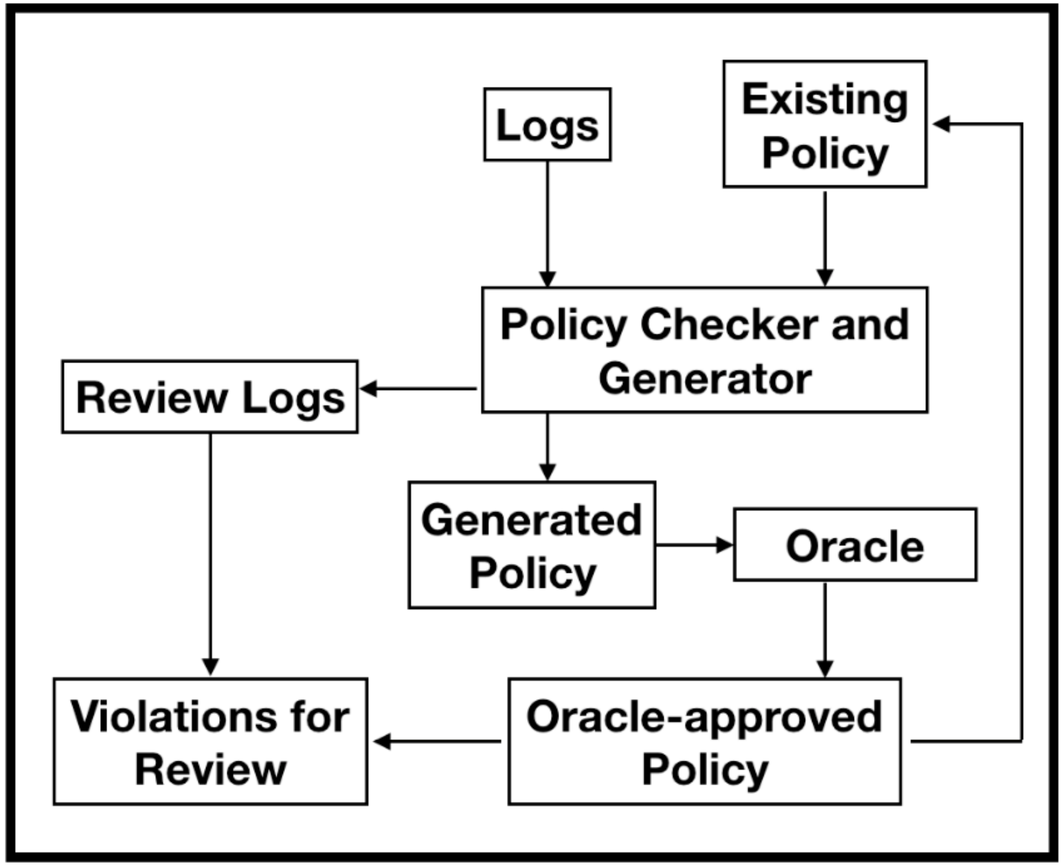
Policy Inference and Audit Framework



Policy Inference and Audit Framework



Policy Inference and Audit Framework



Algorithm for Policy Inference and Audit

Input: $\mathcal{L}$ - a list of log entries
φ_o - existing policy
Σ - typing environment
R - relational database
Output: φ_n - inferred policy
M - map from formula to set of formulas
Function Generate ($\mathcal{L}$, φ_o, Σ, R) **:**

```
1   | φ_o ← infer(L, R, Σ); φ_n ← φ_o; M ← [];
2   | foreach s_i ∈ φ_o do
3   |   | foreach s_j ∈ φ_o \ {s_i} do
4   |   |   | if reduce(s_i, s_j, Σ) then
5   |   |   |   | φ_n ← φ_n \ s_j;
6   |   |   |   | M ← appendMap(M, s_i, s_j);
7   |   |   | else if reduce(s_j, s_i, Σ) then
8   |   |   |   | φ_n ← φ_n \ s_i;
9   |   |   |   | M ← appendMap(M, s_j, s_i);
10  |   |   |   | break;
11  |   |   | end
12  |   | end
13  | end
```

End Function

Algorithm for Policy Inference and Audit

Input: $\mathcal{L}$ - a list of log entries
φ_o - existing policy
Σ - typing environment
R - relational database

Output: φ_n - inferred policy
M - map from formula to set of formulas

Function Generate ($\mathcal{L}$, φ_o, Σ, R) **:**

$\varphi_o \leftarrow \text{infer}(\mathcal{L}, R, \Sigma)$; $\varphi_n \leftarrow \varphi_o$; $M \leftarrow []$;
foreach $s_i \in \varphi_o$ **do**
  foreach $s_j \in \varphi_o \setminus \{s_i\}$ **do**
    if reduce(s_i, s_j, Σ) **then**
      $\varphi_n \leftarrow \varphi_n \setminus s_j$;
      $M \leftarrow \text{appendMap}(M, s_i, s_j)$;
    else if reduce(s_j, s_i, Σ) **then**
      $\varphi_n \leftarrow \varphi_n \setminus s_i$;
      $M \leftarrow \text{appendMap}(M, s_j, s_i)$;
      break;
    end
  end
end
End Function

Algorithm for Policy Inference and Audit

Input: $\mathcal{L}$ - a list of log entries
φ_o - existing policy
Σ - typing environment
R - relational database

Output: φ_n - inferred policy
M - map from formula to set of formulas

Function Generate($\mathcal{L}$, φ_o, Σ, R):

$\varphi_o \leftarrow \text{infer}(\mathcal{L}, R, \Sigma)$; $\varphi_n \leftarrow \varphi_o$; $M \leftarrow []$;
foreach $s_i \in \varphi_o$ **do**
  foreach $s_j \in \varphi_o \setminus \{s_i\}$ **do**
    if reduce(s_i, s_j, Σ) **then**
      $\varphi_n \leftarrow \varphi_n \setminus s_j$;
      $M \leftarrow \text{appendMap}(M, s_i, s_j)$;
    else if reduce(s_j, s_i, Σ) **then**
      $\varphi_n \leftarrow \varphi_n \setminus s_i$;
      $M \leftarrow \text{appendMap}(M, s_j, s_i)$;
      break;
    end
  end
end
End Function

Algorithm for Policy Inference and Audit

Input: $\mathcal{L}$ - a list of log entries
φ_o - existing policy
Σ - typing environment
R - relational database
Output: φ_n - inferred policy
M - map from formula to set of formulas
Function Generate ($\mathcal{L}$, φ_o, Σ, R) **:**

1 $\varphi_o \leftarrow \mathsf{infer}(\mathcal{L}, R, \Sigma)$; $\varphi_n \leftarrow \varphi_o$; $M \leftarrow []$;
2 **foreach** $s_i \in \varphi_o$ **do**
3 **foreach** $s_j \in \varphi_o \setminus \{s_i\}$ **do**
4 **if** $\mathsf{reduce}(s_i, s_j, \Sigma)$ **then**
5 $\varphi_n \leftarrow \varphi_n \setminus s_j$;
6 $M \leftarrow \mathsf{appendMap}(M, s_i, s_j)$;
7 **else if** $\mathsf{reduce}(s_j, s_i, \Sigma)$ **then**
8 $\varphi_n \leftarrow \varphi_n \setminus s_i$;
9 $M \leftarrow \mathsf{appendMap}(M, s_j, s_i)$;
10 break;
11 **end**
12 **end**
13 **end**
End Function

Algorithm for Policy Inference and Audit

Input: $\mathcal{L}$ - a list of log entries
φ_o - existing policy
Σ - typing environment
R - relational database
Output: φ_n - inferred policy
M - map from formula to set of formulas
Function Generate ($\mathcal{L}$, φ_o, Σ, R) **:**

```
1   | φ_o ← infer(L, R, Σ); φ_n ← φ_o; M ← [];
2   | foreach s_i ∈ φ_o do
3   |   | foreach s_j ∈ φ_o \ {s_i} do
4   |   |   | if reduce(s_i, s_j, Σ) then
5   |   |   |   | φ_n ← φ_n \ s_j;
6   |   |   |   | M ← appendMap(M, s_i, s_j);
7   |   |   | else if reduce(s_j, s_i, Σ) then
8   |   |   |   | φ_n ← φ_n \ s_i;
9   |   |   |   | M ← appendMap(M, s_j, s_i);
10  |   |   |   | break;
11  |   |   | end
12  |   | end
13  | end
```

End Function

Algorithm for Policy Inference and Audit

Input: $\mathcal{L}$ - a list of log entries
φ_o - existing policy
Σ - typing environment
R - relational database
Output: φ_n - inferred policy
M - map from formula to set of formulas
Function Generate ($\mathcal{L}$, φ_o, Σ, R) **:**

```
1   φ_o ← infer(L, R, Σ); φ_n ← φ_o; M ← [];
2   foreach s_i ∈ φ_o do
3       foreach s_j ∈ φ_o \ {s_i} do
4           if reduce(s_i, s_j, Σ) then
5               φ_n ← φ_n \ s_j;
6               M ← appendMap(M, s_i, s_j);
7           else if reduce(s_j, s_i, Σ) then
8               φ_n ← φ_n \ s_i;
9               M ← appendMap(M, s_j, s_i);
10              break;
11          end
12      end
13  end
```

End Function

Algorithm for Policy Inference and Audit

Input: $\mathcal{L}$ - a list of log entries
φ_o - existing policy
Σ - typing environment
R - relational database
Output: φ_n - inferred policy
M - map from formula to set of formulas
Function Generate ($\mathcal{L}$, φ_o, Σ, R) **:**

```
1   | φ_o ← infer(L, R, Σ); φ_n ← φ_o; M ← [];
2   | foreach s_i ∈ φ_o do
3   |   | foreach s_j ∈ φ_o \ {s_i} do
4   |   |   | if reduce(s_i, s_j, Σ) then
5   |   |   |   | φ_n ← φ_n \ s_j;
6   |   |   |   | M ← appendMap(M, s_i, s_j);
7   |   |   | else if reduce(s_j, s_i, Σ) then
8   |   |   |   | φ_n ← φ_n \ s_i;
9   |   |   |   | M ← appendMap(M, s_j, s_i);
10  |   |   |   | break;
11  |   |   | end
12  |   | end
13  | end
```

End Function

Algorithm for Policy Inference and Audit

Input: $\mathcal{L}$ - a list of log entries
φ_o - existing policy
Σ - typing environment
R - relational database
Output: φ_n - inferred policy
M - map from formula to set of formulas
Function Generate $(\mathcal{L},\ \varphi_o,\ \Sigma,\ R)$ **:**

```
1   φ_o ← infer(L, R, Σ); φ_n ← φ_o; M ← [];
2   foreach s_i ∈ φ_o do
3       foreach s_j ∈ φ_o \ {s_i} do
4           if reduce(s_i, s_j, Σ) then
5               φ_n ← φ_n \ s_j;
6               M ← appendMap(M, s_i, s_j);
7           else if reduce(s_j, s_i, Σ) then
8               φ_n ← φ_n \ s_i;
9               M ← appendMap(M, s_j, s_i);
10              break;
11          end
12      end
13  end
```

End Function

Input: φ - policy
M - map of formulas to list of formulas
Output: φ_a - approved policy
Function oracle $(\varphi,\ M)$ **:**

```
1   φ_a ← {}; φ_m ← {};
2   foreach s ∈ φ do
3       if isValid(s) then
4           φ_a ← {s} ∪ φ_a;
5       else
6           φ_s ← M(s) ∪ φ_s;
7       end
8   end
9   foreach s ∈ φ_s do
10      if isValid(s) then
11          φ_a ← {s} ∪ φ_a;
12      end
13  end
```

End Function

Algorithm for Policy Inference and Audit

Input: $\mathcal{L}$ - a list of log entries
φ_o - existing policy
Σ - typing environment
R - relational database
Output: φ_n - inferred policy
M - map from formula to set of formulas
Function Generate $(\mathcal{L},\ \varphi_o,\ \Sigma,\ R)$ **:**

```
1   φ_o ← infer(L, R, Σ); φ_n ← φ_o; M ← [];
2   foreach s_i ∈ φ_o do
3       foreach s_j ∈ φ_o \ {s_i} do
4           if reduce(s_i, s_j, Σ) then
5               φ_n ← φ_n \ s_j;
6               M ← appendMap(M, s_i, s_j);
7           else if reduce(s_j, s_i, Σ) then
8               φ_n ← φ_n \ s_i;
9               M ← appendMap(M, s_j, s_i);
10              break;
11          end
12      end
13  end
```

End Function

Input: φ - policy
M - map of formulas to list of formulas
Output: φ_a - approved policy
Function oracle $(\varphi,\ M)$ **:**

```
1   φ_a ← {}; φ_m ← {};
2   foreach s ∈ φ do
3       if isValid(s) then
4           φ_a ← {s} ∪ φ_a;
5       else
6           φ_s ← M(s) ∪ φ_s;
7       end
8   end
9   foreach s ∈ φ_s do
10      if isValid(s) then
11          φ_a ← {s} ∪ φ_a;
12      end
13  end
```

End Function

Algorithm for Policy Inference and Audit

Input: $\mathcal{L}$ - a list of log entries
φ_o - existing policy
Σ - typing environment
R - relational database
Output: φ_n - inferred policy
M - map from formula to set of formulas
Function Generate $(\mathcal{L},\ \varphi_o,\ \Sigma,\ R)$ **:**

```
1   φ_o ← infer(L, R, Σ); φ_n ← φ_o; M ← [];
2   foreach s_i ∈ φ_o do
3       foreach s_j ∈ φ_o \ {s_i} do
4           if reduce(s_i, s_j, Σ) then
5               φ_n ← φ_n \ s_j;
6               M ← appendMap(M, s_i, s_j);
7           else if reduce(s_j, s_i, Σ) then
8               φ_n ← φ_n \ s_i;
9               M ← appendMap(M, s_j, s_i);
10              break;
11          end
12      end
13  end
```

End Function

Input: φ - policy
M - map of formulas to list of formulas
Output: φ_a - approved policy
Function oracle $(\varphi,\ M)$ **:**

```
1   φ_a ← {}; φ_m ← {};
2   foreach s ∈ φ do
3       if isValid(s) then
4           φ_a ← {s} ∪ φ_a;
5       else
6           φ_s ← M(s) ∪ φ_s;
7       end
8   end
9   foreach s ∈ φ_s do
10      if isValid(s) then
11          φ_a ← {s} ∪ φ_a;
12      end
13  end
```

End Function

Algorithm for Policy Inference and Audit

Input: $\mathcal{L}$ - a list of log entries
φ_o - existing policy
Σ - typing environment
R - relational database
Output: φ_n - inferred policy
M - map from formula to set of formulas
Function Generate $(\mathcal{L},\ \varphi_o,\ \Sigma,\ R)$ **:**

```
1  | φ_o ← infer(L, R, Σ); φ_n ← φ_o; M ← [];
2  | foreach s_i ∈ φ_o do
3  |   | foreach s_j ∈ φ_o \ {s_i} do
4  |   |   | if reduce(s_i, s_j, Σ) then
5  |   |   |   | φ_n ← φ_n \ s_j;
6  |   |   |   | M ← appendMap(M, s_i, s_j);
7  |   |   | else if reduce(s_j, s_i, Σ) then
8  |   |   |   | φ_n ← φ_n \ s_i;
9  |   |   |   | M ← appendMap(M, s_j, s_i);
10 |   |   |   | break;
11 |   |   | end
12 |   | end
13 | end
End Function
```

Input: φ - policy
M - map of formulas to list of formulas
Output: φ_a - approved policy
Function oracle $(\varphi,\ M)$ **:**

```
1  | φ_a ← {}; φ_m ← {};
2  | foreach s ∈ φ do
3  |   | if isValid(s) then
4  |   |   | φ_a ← {s} ∪ φ_a;
5  |   | else
6  |   |   | φ_s ← M(s) ∪ φ_s;
7  |   | end
8  | end
9  | foreach s ∈ φ_s do
10 |   | if isValid(s) then
11 |   |   | φ_a ← {s} ∪ φ_a;
12 |   | end
13 | end
End Function
```

Algorithm for Policy Inference and Audit

Input: $\mathcal{L}$ - a list of log entries
φ_o - existing policy
Σ - typing environment
R - relational database
Output: φ_n - inferred policy
M - map from formula to set of formulas
Function Generate $(\mathcal{L},\ \varphi_o,\ \Sigma,\ R)$ **:**

```
1   φ_o ← infer(L, R, Σ); φ_n ← φ_o; M ← [];
2   foreach s_i ∈ φ_o do
3       foreach s_j ∈ φ_o \ {s_i} do
4           if reduce(s_i, s_j, Σ) then
5               φ_n ← φ_n \ s_j;
6               M ← appendMap(M, s_i, s_j);
7           else if reduce(s_j, s_i, Σ) then
8               φ_n ← φ_n \ s_i;
9               M ← appendMap(M, s_j, s_i);
10              break;
11          end
12      end
13  end
```

End Function

Input: φ - policy
M - map of formulas to list of formulas
Output: φ_a - approved policy
Function oracle $(\varphi,\ M)$ **:**

```
1   φ_a ← {}; φ_m ← {};
2   foreach s ∈ φ do
3       if isValid(s) then
4           φ_a ← {s} ∪ φ_a;
5       else
6           φ_s ← M(s) ∪ φ_s;
7       end
8   end
9   foreach s ∈ φ_s do
10      if isValid(s) then
11          φ_a ← {s} ∪ φ_a;
12      end
13  end
```

End Function

Algorithm for Policy Inference and Audit

Input: $\mathcal{L}$ - a list of log entries
φ_o - existing policy
Σ - typing environment
R - relational database
Output: φ_n - inferred policy
M - map from formula to set of formulas
Function Generate $(\mathcal{L},\ \varphi_o,\ \Sigma,\ R)$ **:**

```
1   φ_o ← infer(L, R, Σ); φ_n ← φ_o; M ← [];
2   foreach s_i ∈ φ_o do
3       foreach s_j ∈ φ_o \ {s_i} do
4           if reduce(s_i, s_j, Σ) then
5               φ_n ← φ_n \ s_j;
6               M ← appendMap(M, s_i, s_j);
7           else if reduce(s_j, s_i, Σ) then
8               φ_n ← φ_n \ s_i;
9               M ← appendMap(M, s_j, s_i);
10              break;
11          end
12      end
13  end
```

End Function

Input: φ - policy
M - map of formulas to list of formulas
Output: φ_a - approved policy
Function oracle $(\varphi,\ M)$ **:**

```
1   φ_a ← {}; φ_m ← {};
2   foreach s ∈ φ do
3       if isValid(s) then
4           φ_a ← {s} ∪ φ_a;
5       else
6           φ_s ← M(s) ∪ φ_s;
7       end
8   end
9   foreach s ∈ φ_s do
10      if isValid(s) then
11          φ_a ← {s} ∪ φ_a;
12      end
13  end
```

End Function

Algorithm for Policy Inference and Audit

Input: $\mathcal{L}$ - a list of log entries
φ_o - existing policy
Σ - typing environment
R - relational database
Output: φ_n - inferred policy
M - map from formula to set of formulas
Function Generate $(\mathcal{L},\ \varphi_o,\ \Sigma,\ R)$ **:**

```
1   φ_o ← infer(L, R, Σ); φ_n ← φ_o; M ← [];
2   foreach s_i ∈ φ_o do
3       foreach s_j ∈ φ_o \ {s_i} do
4           if reduce(s_i, s_j, Σ) then
5               φ_n ← φ_n \ s_j;
6               M ← appendMap(M, s_i, s_j);
7           else if reduce(s_j, s_i, Σ) then
8               φ_n ← φ_n \ s_i;
9               M ← appendMap(M, s_j, s_i);
10              break;
11          end
12      end
13  end
```

End Function

Input: φ - policy
M - map of formulas to list of formulas
Output: φ_a - approved policy
Function oracle $(\varphi,\ M)$ **:**

```
1   φ_a ← {}; φ_m ← {};
2   foreach s ∈ φ do
3       if isValid(s) then
4           φ_a ← {s} ∪ φ_a;
5       else
6           φ_s ← M(s) ∪ φ_s;
7       end
8   end
9   foreach s ∈ φ_s do
10      if isValid(s) then
11          φ_a ← {s} ∪ φ_a;
12      end
13  end
```

End Function

Properties of Inference

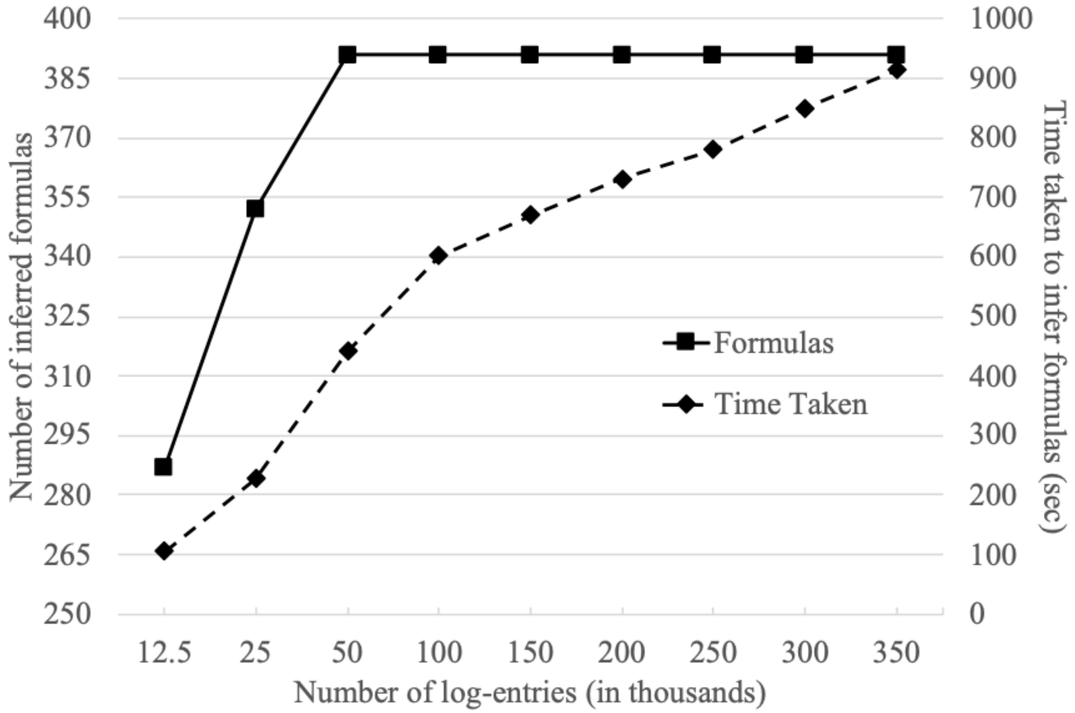
- Monotonicity
  - For a fixed set of relations, adding more entries to a log with an inferred policy results in a more or equally permissive policy.
- Termination
  - Each iteration of the algorithm terminates.
- Soundness
  - The log from which our algorithm infers a policy satisfies the inferred policy.
- Minimality
  - The policy inferred by our algorithm is the most restrictive policy (that can be inferred by our algorithm) that the log satisfies.

Prototype Evaluation

- Implemented the algorithm in Python
  - Log is a table in the database along with the different attributes of data
  - Assumes only one resource is accessed per entry
  - Fixed attributes and relationships between data and principals
  - Employ human oracle for validation
- Generate simulated logs for healthcare organization
  - Based on HIPAA (prior work by Garg et al. [CCS ’11])

Prototype Evaluation

- Time taken and the number of formulas inferred for different no. of log entries



Summary - Automating Audit with Policy Inference

- Infer policies from logs and audit policies
  - Reduce number of entries to audit
  - Reuse policies for enforcement