



Automating Audit with Policy Inference

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

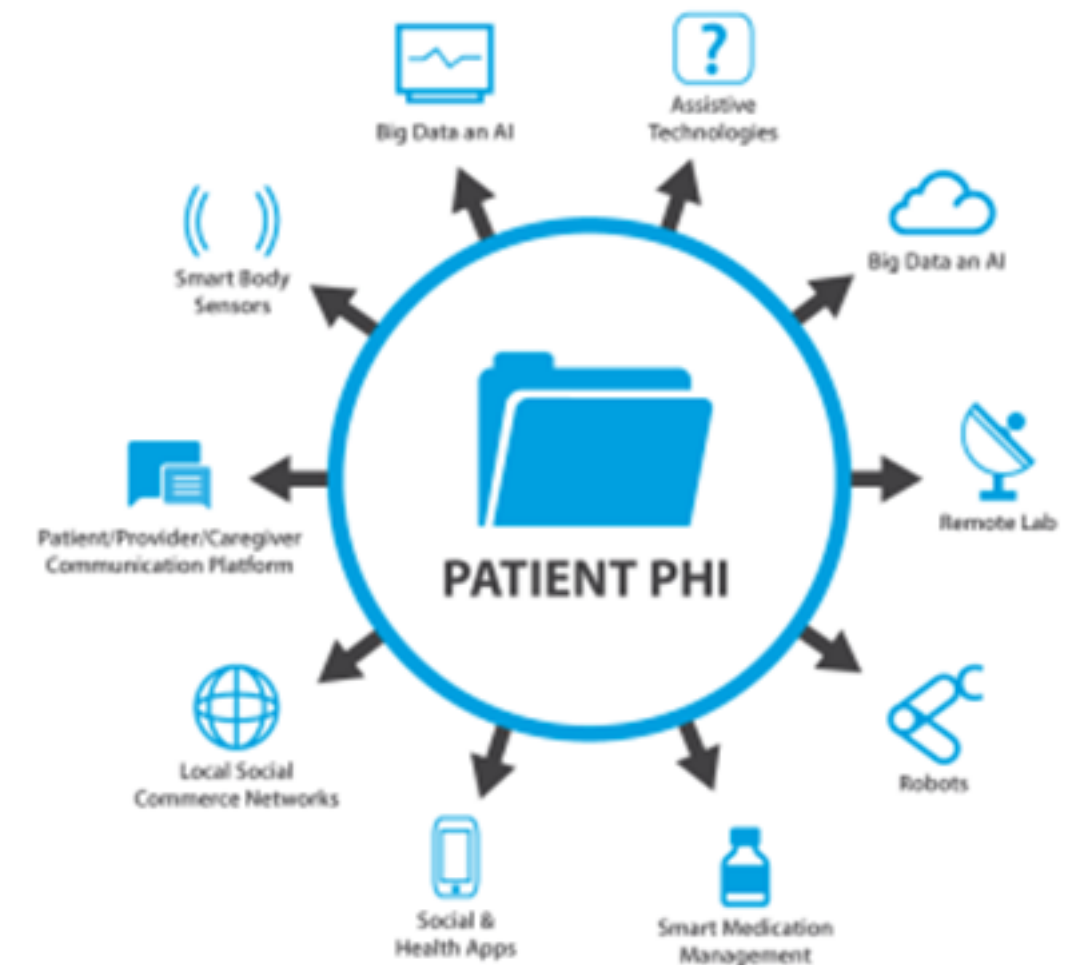
Organizational Data Security



=



=



=



Organizational Data Security



Google Hangouts App Resulting In Messages Sent To Wrong Recipients

DARRELL ETHERINGTON

Thursday, September 26th, 2013



Updated: Google says they've apparently fixed its cause.

A number of sources, including tipsters, forum posters and individuals on Twitter are reporting trouble routing instant messages correctly.

Facebook Apps Lead To Data Breach [REPORT]

4.9k / 2.2k 2.1k 0 359

Ads by Google
[Fahren nach Dänemark 2013](#) - Aalen, Bornholm und mehr Z
[Berufliche Weiterbildung](#) - Nebenberufliche Weiterbildung



BY STAN SCHRÖDENER
Some Facebook applications... to third party companies... The apps in question have... of the 10 most popular Facebook... specifically internet res...

TECHNOLOGY

Facebook Bug Exposed Numbers Of 6 Million

06/21/2013 07:04 pm ET | Updated Jun 22, 2013

3.5k f t p in

Alexis Kleinman
Deputy Managing Editor Of Impact & Innovation



Healthcare IT News

GLOBAL EDITION

Employee error exposed data of 16,000 Blue Cross patients online for 3 months

An employee uploaded a file containing member information to a public-facing website in April, but officials did not discover the error until July.

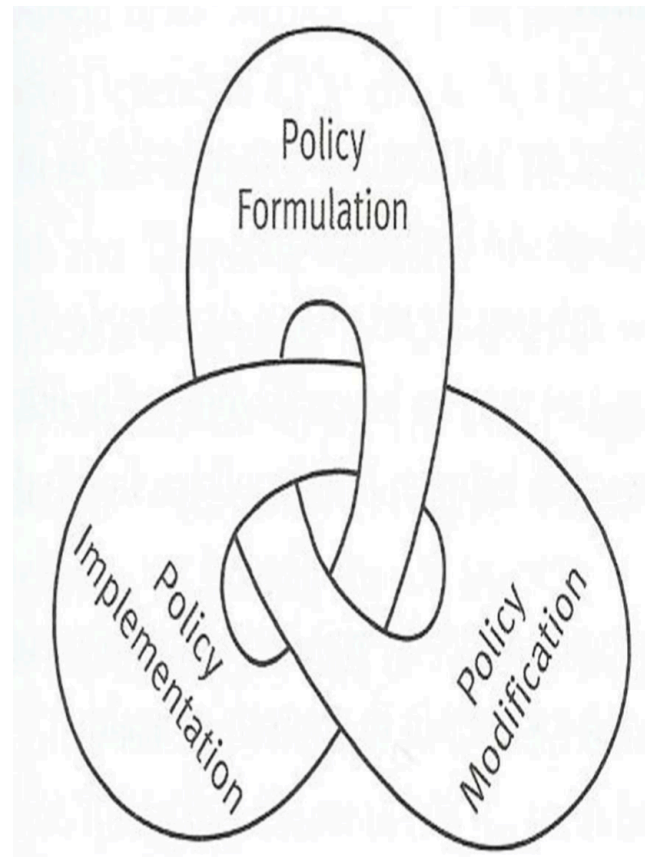
By **Jessica Davis** | September 21, 2018 | 09:30 AM

f t in

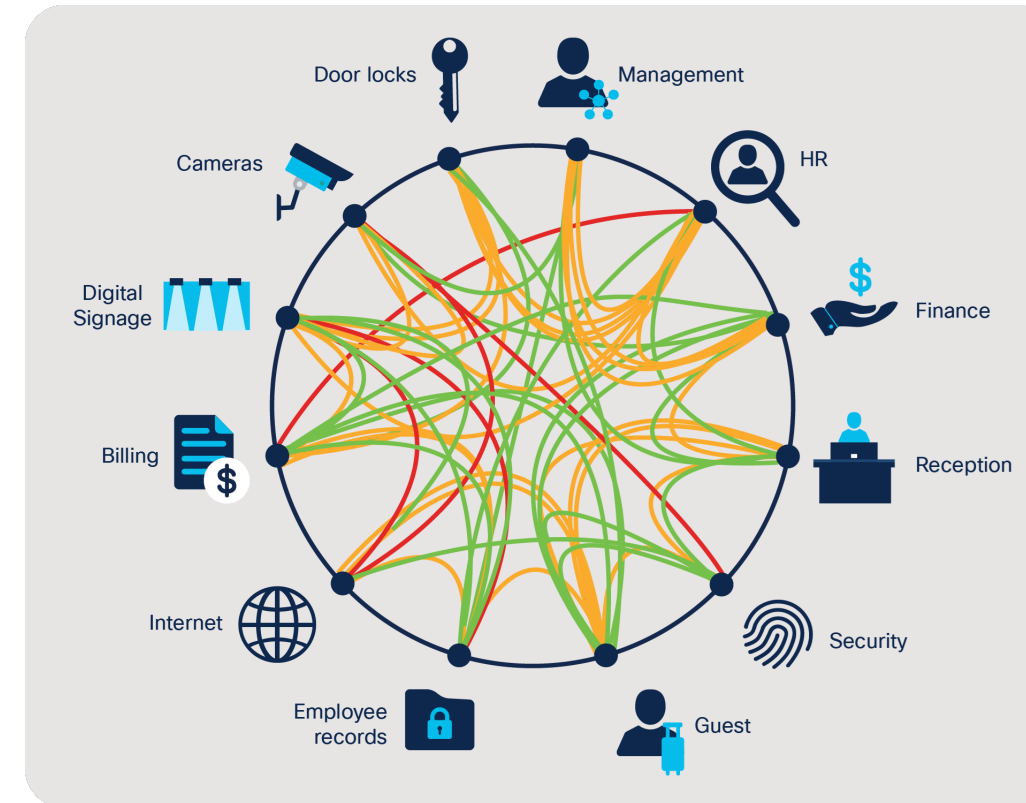


Factors Affecting Policy Enforcement

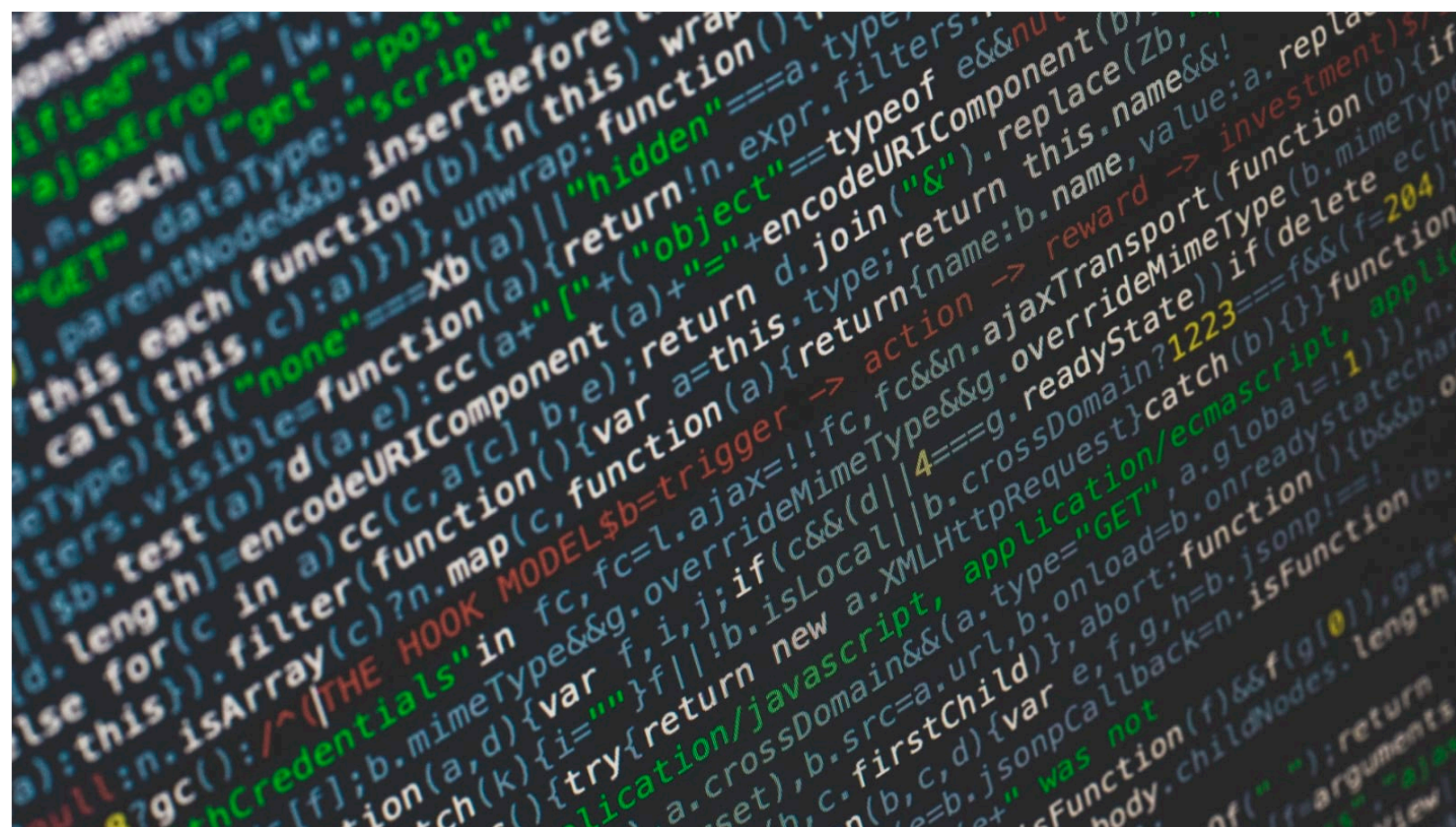
Constant policy modification



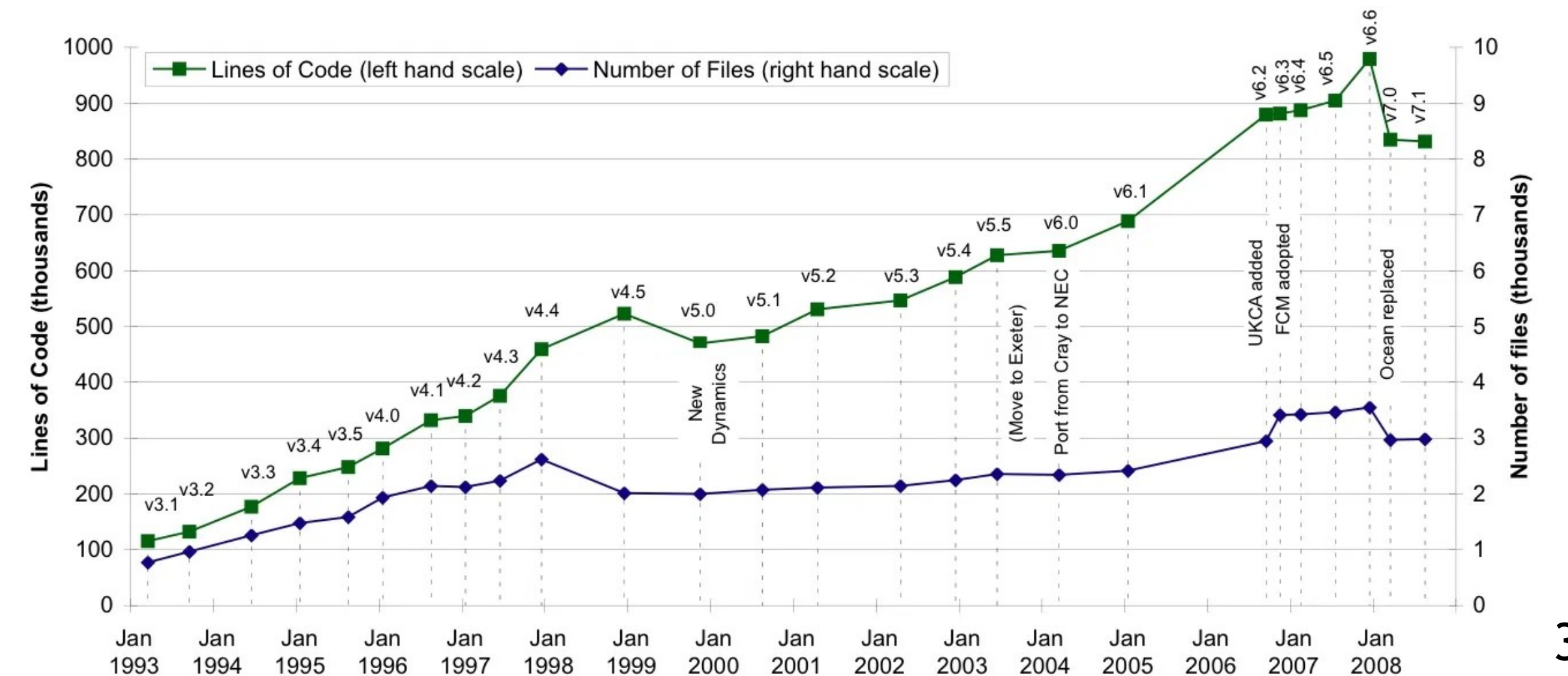
Complexity of policy



Size of the codebase



Evolution of code



Factors Affecting Policy Enforcement

- Organizations may lack explicit, mechanized policy specification



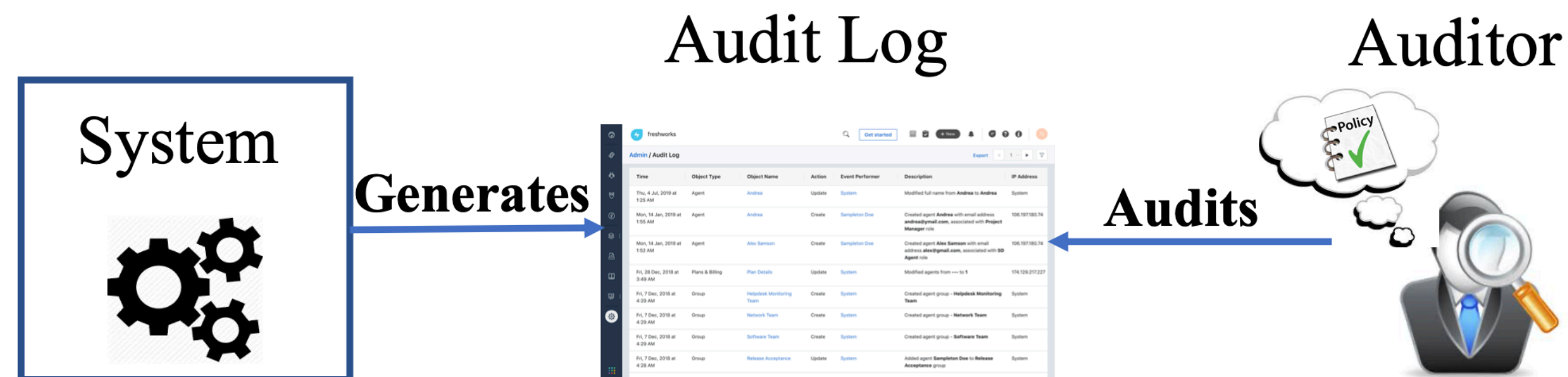
- Inline checks are error-prone; introduce bugs

```
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



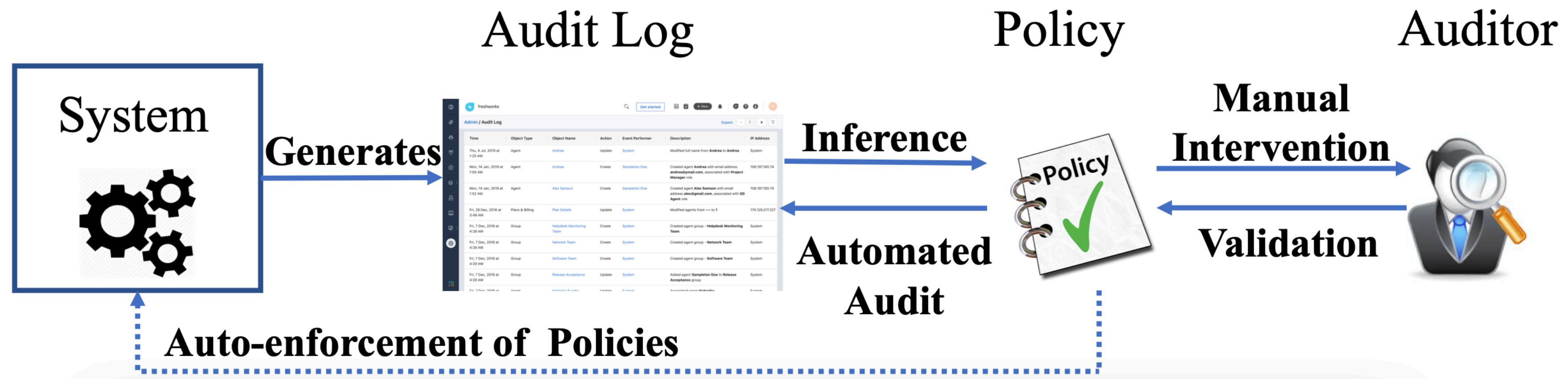
Our Approach - Insight and Overview

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

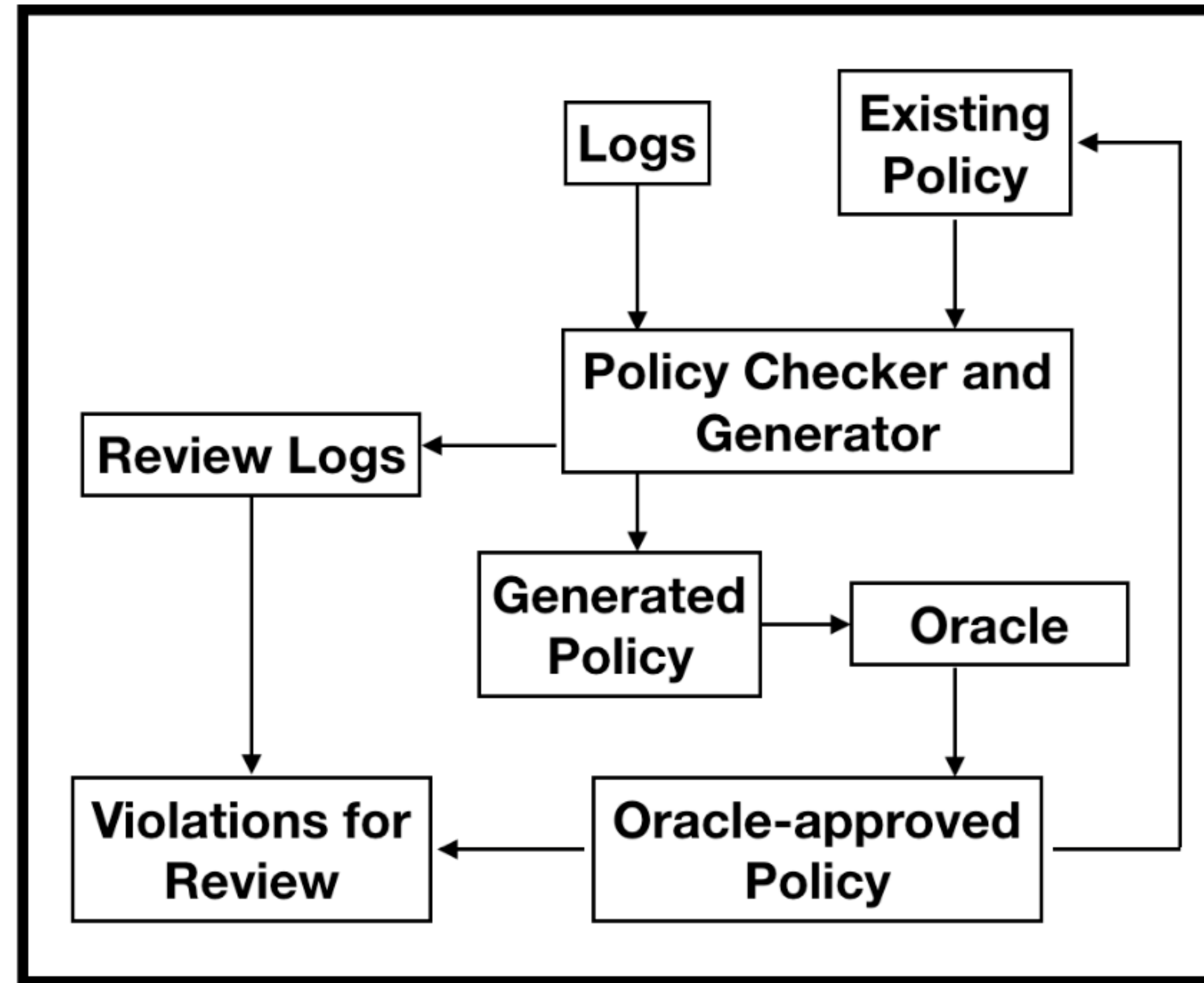


Our Approach - Insight and Overview

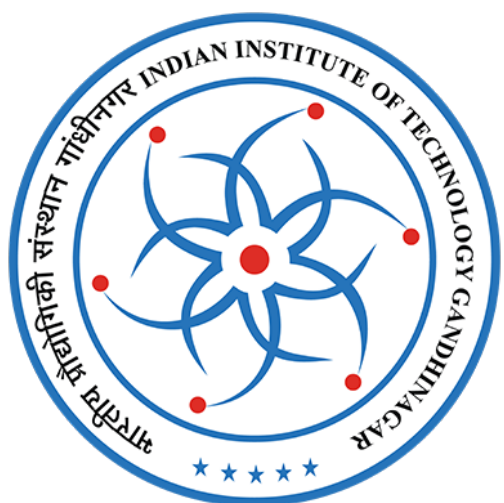
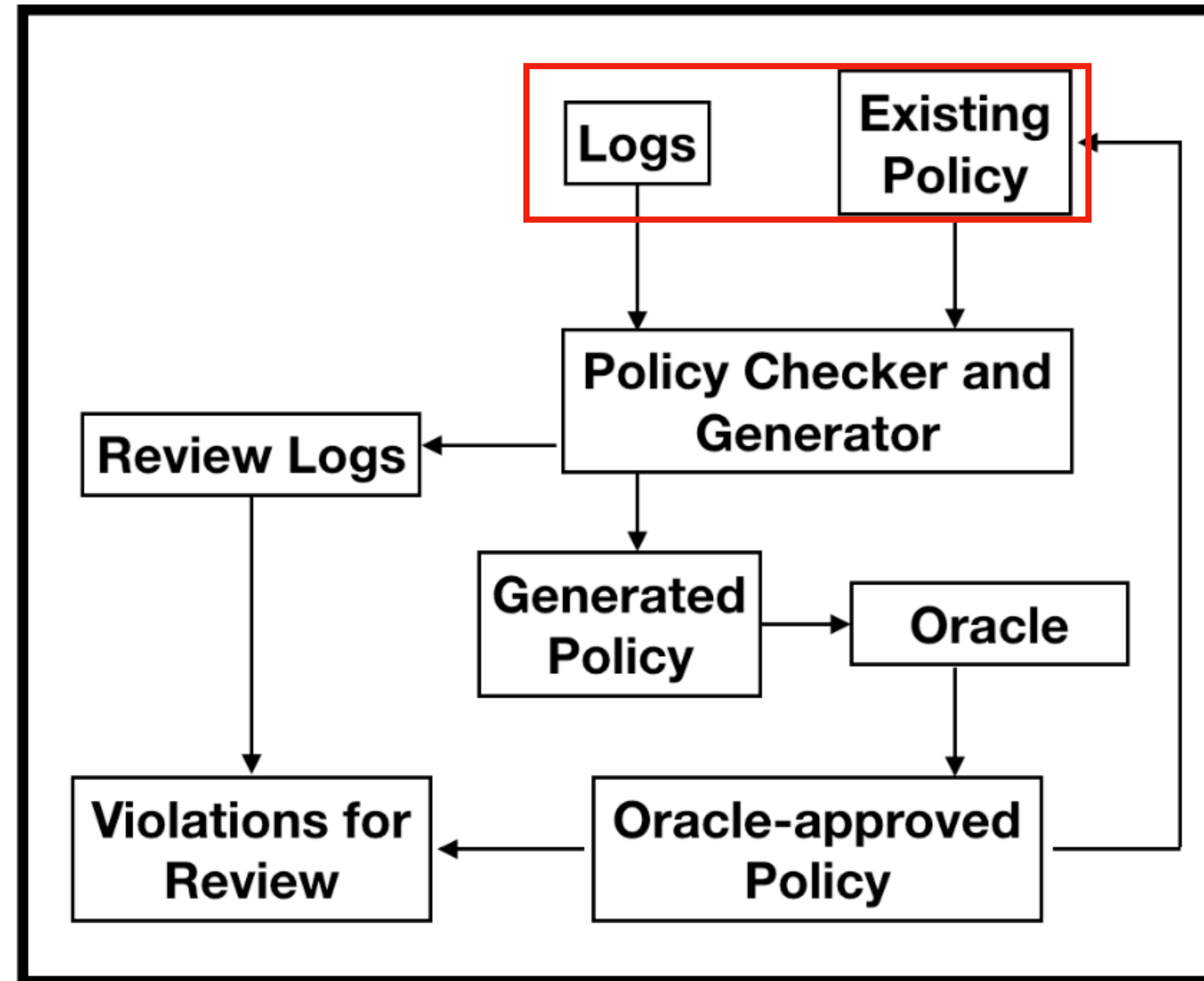
Audit logs combined with auditor-decisions, are sufficient to reconstruct a correct mechanized policy, which can be used to reduce the cost of manually auditing logs



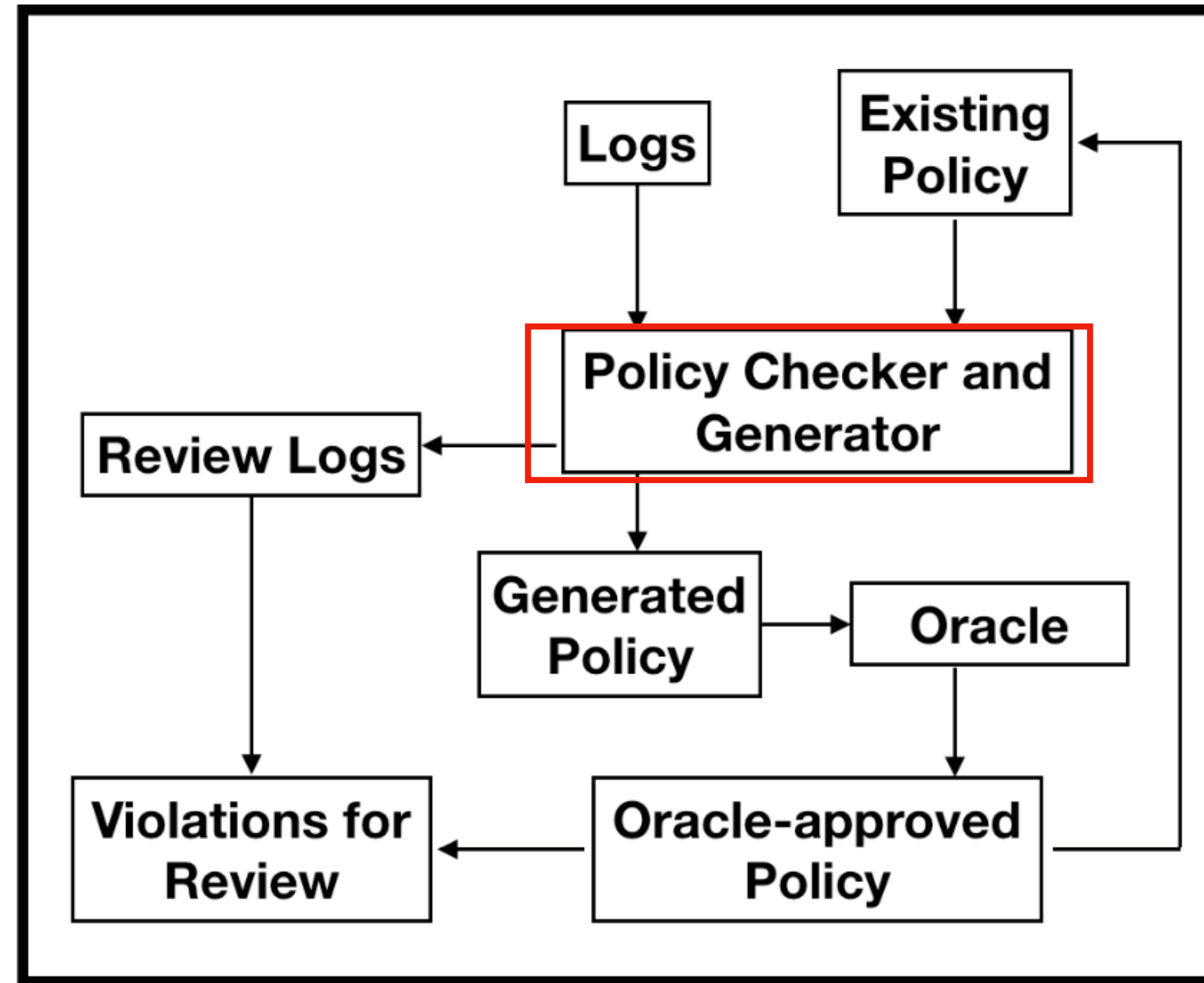
Policy Inference and Audit Framework



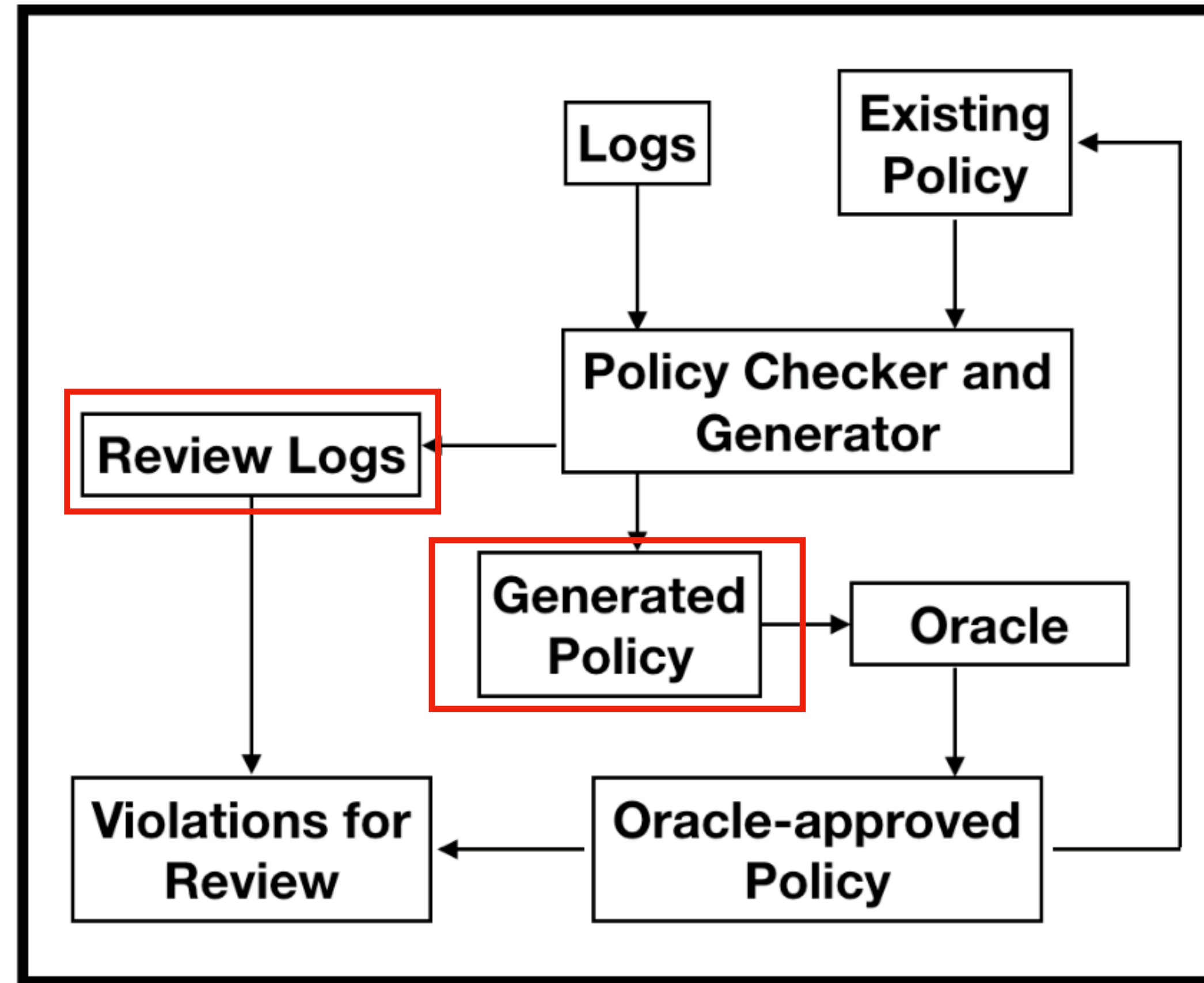
Policy Inference and Audit Framework



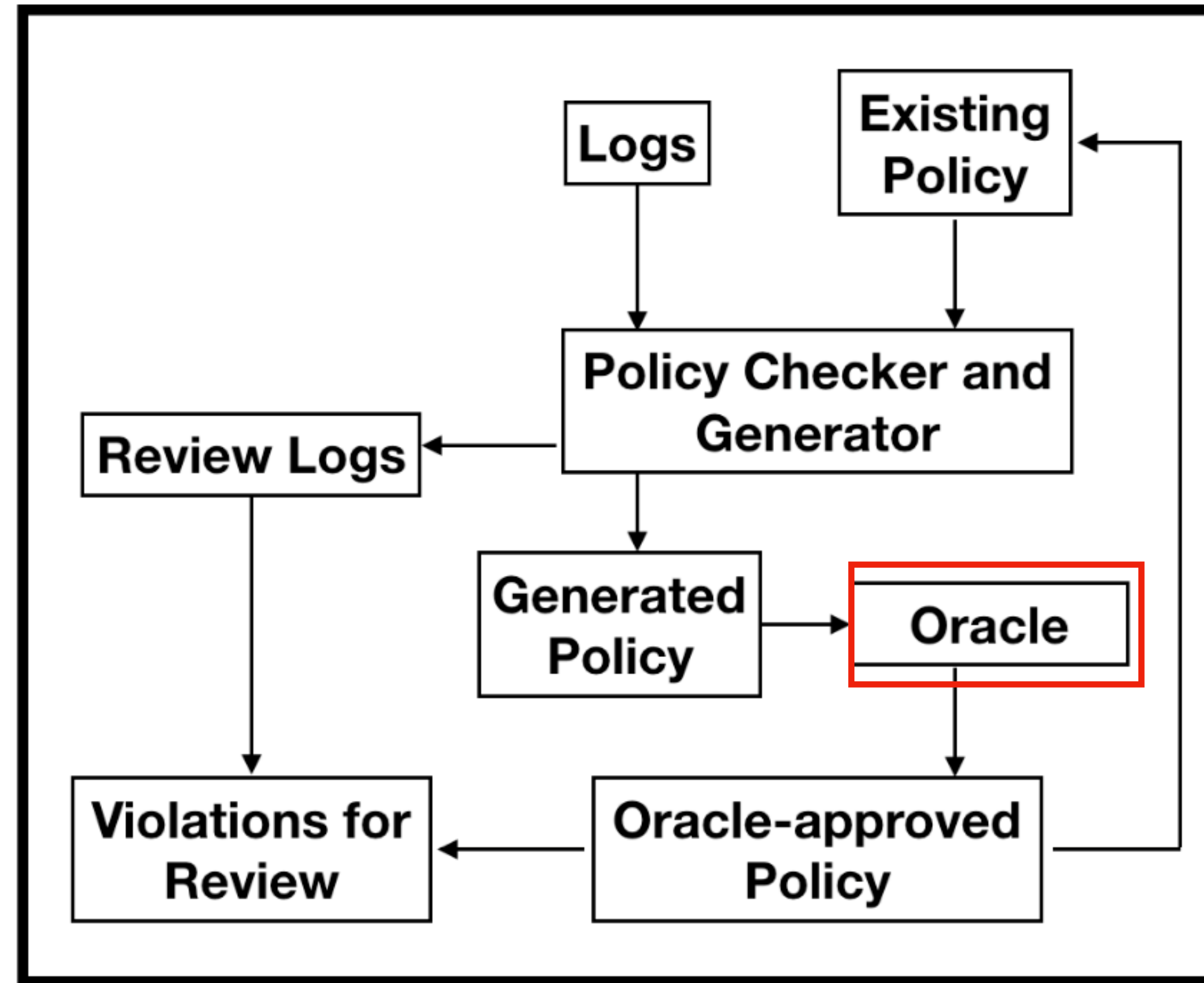
Policy Inference and Audit Framework



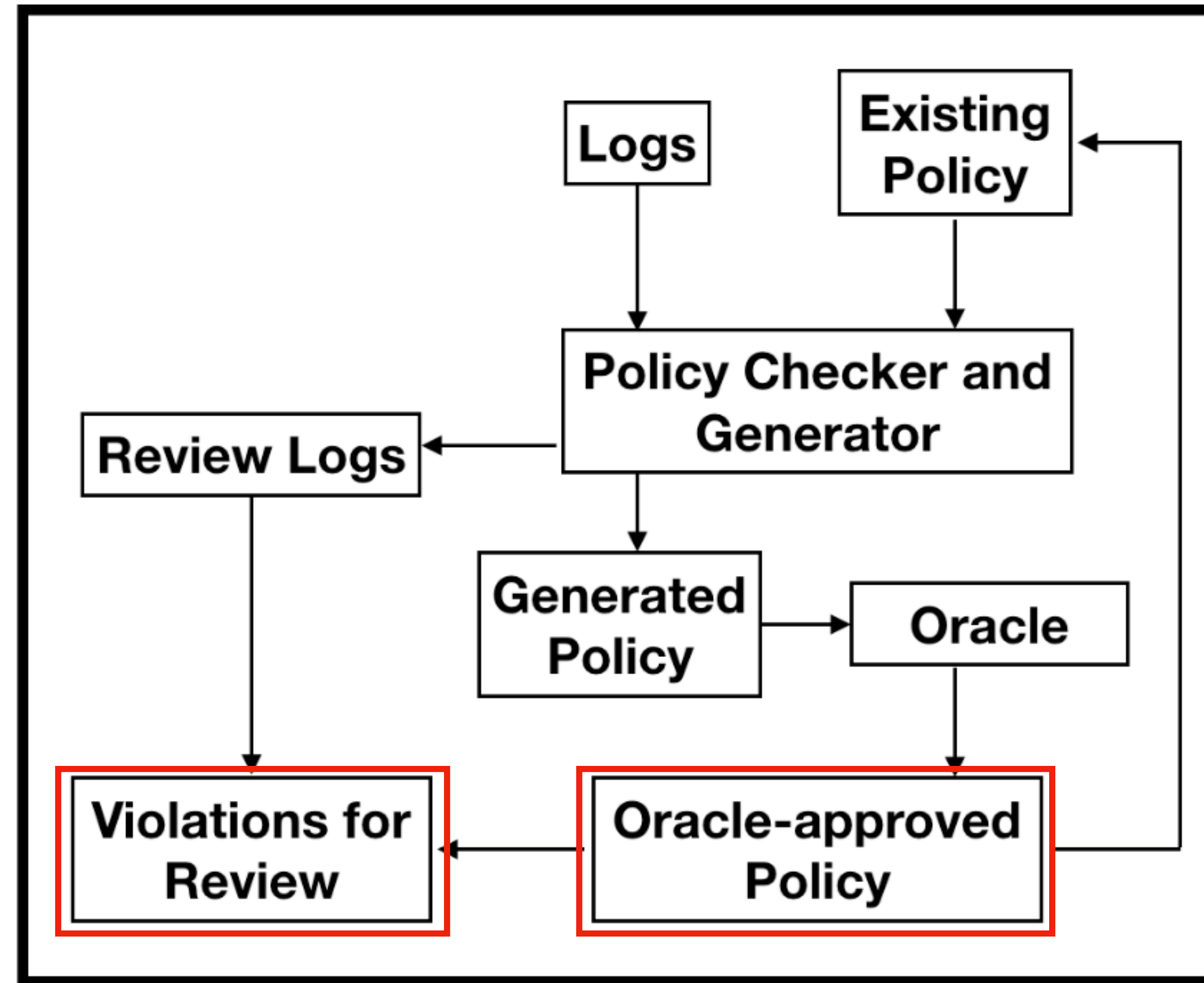
Policy Inference and Audit Framework



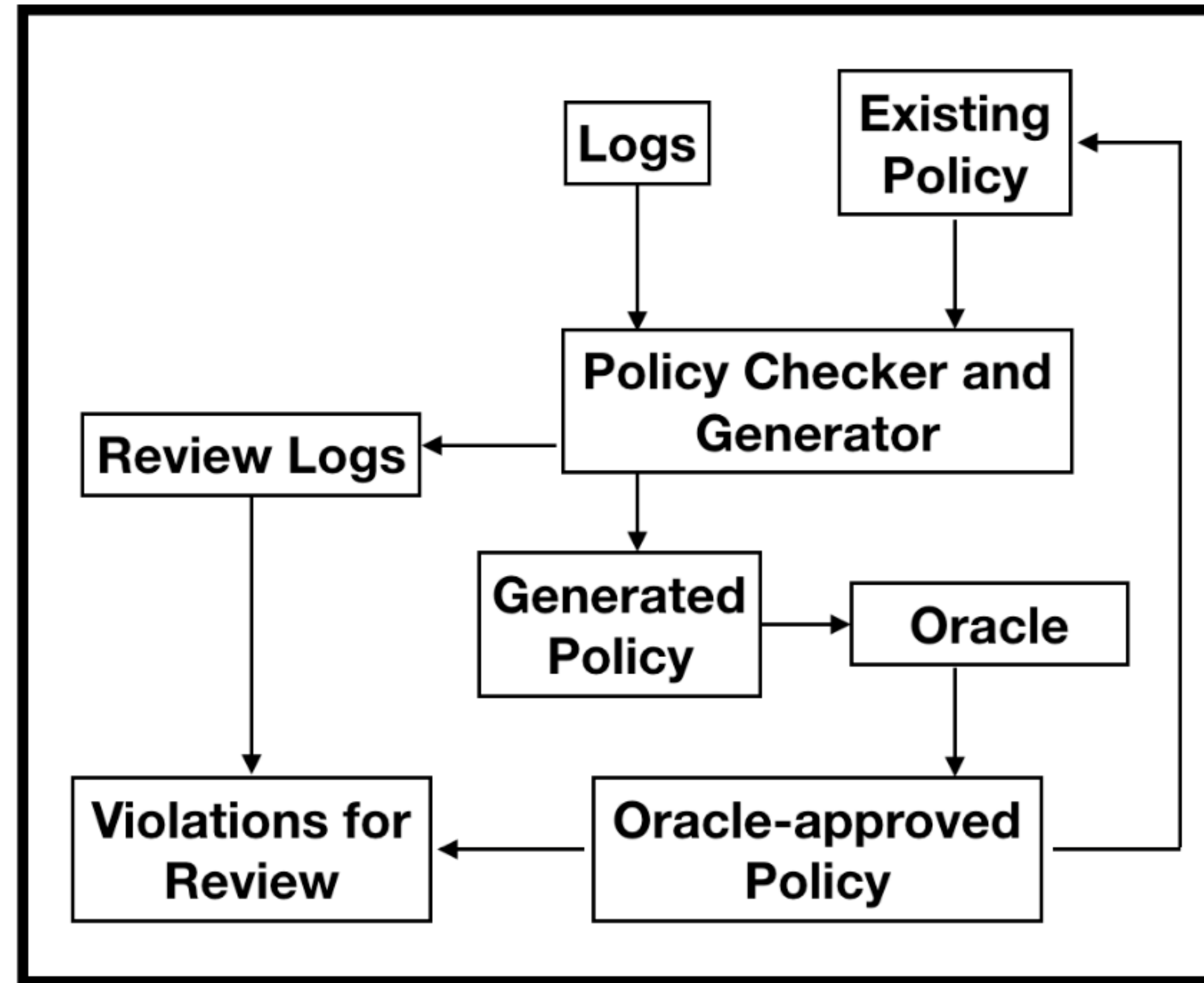
Policy Inference and Audit Framework



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($\mathcal{L}, \varphi_o, \Sigma, R$):

```
1   $\varphi_o \leftarrow \text{infer}(\mathcal{L}, R, \Sigma); \varphi_n \leftarrow \varphi_o; M \leftarrow [];$ 
2  foreach  $s_i \in \varphi_o$  do
3      foreach  $s_j \in \varphi_o \setminus \{s_i\}$  do
4          if  $\text{reduce}(s_i, s_j, \Sigma)$  then
5               $\varphi_n \leftarrow \varphi_n \setminus s_j;$ 
6               $M \leftarrow \text{appendMap}(M, s_i, s_j);$ 
7          else if  $\text{reduce}(s_j, s_i, \Sigma)$  then
8               $\varphi_n \leftarrow \varphi_n \setminus s_i;$ 
9               $M \leftarrow \text{appendMap}(M, s_j, s_i);$ 
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   $\varphi_o \leftarrow \text{infer}(\mathcal{L}, R, \Sigma); \varphi_n \leftarrow \varphi_o; M \leftarrow [];$   
2  foreach  $s_i \in \varphi_o$  do  
3      foreach  $s_j \in \varphi_o \setminus \{s_i\}$  do  
4          if reduce( $s_i, s_j, \Sigma$ ) then  
5               $\varphi_n \leftarrow \varphi_n \setminus s_j;$   
6               $M \leftarrow \text{appendMap}(M, s_i, s_j);$   
7          else if reduce( $s_j, s_i, \Sigma$ ) then  
8               $\varphi_n \leftarrow \varphi_n \setminus s_i;$   
9               $M \leftarrow \text{appendMap}(M, s_j, s_i);$   
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   $\varphi_o \leftarrow \text{infer}(\mathcal{L}, R, \Sigma); \varphi_n \leftarrow \varphi_o; M \leftarrow [];$   
2  foreach  $s_i \in \varphi_o$  do  
3      foreach  $s_j \in \varphi_o \setminus \{s_i\}$  do  
4          if reduce( $s_i, s_j, \Sigma$ ) then  
5               $\varphi_n \leftarrow \varphi_n \setminus s_j;$   
6               $M \leftarrow \text{appendMap}(M, s_i, s_j);$   
7          else if reduce( $s_j, s_i, \Sigma$ ) then  
8               $\varphi_n \leftarrow \varphi_n \setminus s_i;$   
9               $M \leftarrow \text{appendMap}(M, s_j, s_i);$   
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   $\varphi_o \leftarrow \text{infer}(\mathcal{L}, R, \Sigma); \varphi_n \leftarrow \varphi_o; M \leftarrow [];$ 
2  foreach  $s_i \in \varphi_o$  do
3      foreach  $s_j \in \varphi_o \setminus \{s_i\}$  do
4          if reduce( $s_i, s_j, \Sigma$ ) then
5               $\varphi_n \leftarrow \varphi_n \setminus s_j;$ 
6               $M \leftarrow \text{appendMap}(M, s_i, s_j);$ 
7          else if reduce( $s_j, s_i, \Sigma$ ) then
8               $\varphi_n \leftarrow \varphi_n \setminus s_i;$ 
9               $M \leftarrow \text{appendMap}(M, s_j, s_i);$ 
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   $\varphi_o \leftarrow \text{infer}(\mathcal{L}, R, \Sigma); \varphi_n \leftarrow \varphi_o; M \leftarrow [];$ 
2  foreach  $s_i \in \varphi_o$  do
3      foreach  $s_j \in \varphi_o \setminus \{s_i\}$  do
4          if  $\text{reduce}(s_i, s_j, \Sigma)$  then
5               $\varphi_n \leftarrow \varphi_n \setminus s_j;$ 
6               $M \leftarrow \text{appendMap}(M, s_i, s_j);$ 
7          else if  $\text{reduce}(s_j, s_i, \Sigma)$  then
8               $\varphi_n \leftarrow \varphi_n \setminus s_i;$ 
9               $M \leftarrow \text{appendMap}(M, s_j, s_i);$ 
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   $\varphi_o \leftarrow \text{infer}(\mathcal{L}, R, \Sigma); \varphi_n \leftarrow \varphi_o; M \leftarrow [];$ 
2  foreach  $s_i \in \varphi_o$  do
3      foreach  $s_j \in \varphi_o \setminus \{s_i\}$  do
4          if  $\text{reduce}(s_i, s_j, \Sigma)$  then
5               $\varphi_n \leftarrow \varphi_n \setminus s_j;$ 
6               $M \leftarrow \text{appendMap}(M, s_i, s_j);$ 
7          else if  $\text{reduce}(s_j, s_i, \Sigma)$  then
8               $\varphi_n \leftarrow \varphi_n \setminus s_i;$ 
9               $M \leftarrow \text{appendMap}(M, s_j, s_i);$ 
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   $\varphi_o \leftarrow \text{infer}(\mathcal{L}, R, \Sigma); \varphi_n \leftarrow \varphi_o; M \leftarrow [];$ 
2  foreach  $s_i \in \varphi_o$  do
3      foreach  $s_j \in \varphi_o \setminus \{s_i\}$  do
4          if reduce( $s_i, s_j, \Sigma$ ) then
5               $\varphi_n \leftarrow \varphi_n \setminus s_j;$ 
6               $M \leftarrow \text{appendMap}(M, s_i, s_j);$ 
7          else if reduce( $s_j, s_i, \Sigma$ ) then
8               $\varphi_n \leftarrow \varphi_n \setminus s_i;$ 
9               $M \leftarrow \text{appendMap}(M, s_j, s_i);$ 
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   $\varphi_o \leftarrow \text{infer}(\mathcal{L}, R, \Sigma); \varphi_n \leftarrow \varphi_o; M \leftarrow [];$ 
2  foreach  $s_i \in \varphi_o$  do
3      foreach  $s_j \in \varphi_o \setminus \{s_i\}$  do
4          if reduce( $s_i, s_j, \Sigma$ ) then
5               $\varphi_n \leftarrow \varphi_n \setminus s_j;$ 
6               $M \leftarrow \text{appendMap}(M, s_i, s_j);$ 
7          else if reduce( $s_j, s_i, \Sigma$ ) then
8               $\varphi_n \leftarrow \varphi_n \setminus s_i;$ 
9               $M \leftarrow \text{appendMap}(M, s_j, s_i);$ 
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(φ, M):

```

1   $\varphi_a \leftarrow \{\}; \varphi_m \leftarrow \{\};$ 
2  foreach  $s \in \varphi$  do
3      if isValid( $s$ ) then
4           $\varphi_a \leftarrow \{s\} \cup \varphi_a;$ 
5      else
6           $\varphi_s \leftarrow M(s) \cup \varphi_s;$ 
7      end
8  end
9  foreach  $s \in \varphi_s$  do
10     if isValid( $s$ ) then
11          $\varphi_a \leftarrow \{s\} \cup \varphi_a;$ 
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   $\varphi_o \leftarrow \text{infer}(\mathcal{L}, R, \Sigma); \varphi_n \leftarrow \varphi_o; M \leftarrow [];$ 
2  foreach  $s_i \in \varphi_o$  do
3      foreach  $s_j \in \varphi_o \setminus \{s_i\}$  do
4          if reduce( $s_i, s_j, \Sigma$ ) then
5               $\varphi_n \leftarrow \varphi_n \setminus s_j;$ 
6               $M \leftarrow \text{appendMap}(M, s_i, s_j);$ 
7          else if reduce( $s_j, s_i, \Sigma$ ) then
8               $\varphi_n \leftarrow \varphi_n \setminus s_i;$ 
9               $M \leftarrow \text{appendMap}(M, s_j, s_i);$ 
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(φ, M):

```

1   $\varphi_a \leftarrow \{\}; \varphi_m \leftarrow \{\};$ 
2  foreach  $s \in \varphi$  do
3      if isValid( $s$ ) then
4           $\varphi_a \leftarrow \{s\} \cup \varphi_a;$ 
5      else
6           $\varphi_s \leftarrow M(s) \cup \varphi_s;$ 
7      end
8  end
9  foreach  $s \in \varphi_s$  do
10     if isValid( $s$ ) then
11          $\varphi_a \leftarrow \{s\} \cup \varphi_a;$ 
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   $\varphi_o \leftarrow \text{infer}(\mathcal{L}, R, \Sigma); \varphi_n \leftarrow \varphi_o; M \leftarrow [];$ 
2  foreach  $s_i \in \varphi_o$  do
3      foreach  $s_j \in \varphi_o \setminus \{s_i\}$  do
4          if reduce( $s_i, s_j, \Sigma$ ) then
5               $\varphi_n \leftarrow \varphi_n \setminus s_j;$ 
6               $M \leftarrow \text{appendMap}(M, s_i, s_j);$ 
7          else if reduce( $s_j, s_i, \Sigma$ ) then
8               $\varphi_n \leftarrow \varphi_n \setminus s_i;$ 
9               $M \leftarrow \text{appendMap}(M, s_j, s_i);$ 
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(φ, M):

```

1   $\varphi_a \leftarrow \{\}; \varphi_m \leftarrow \{\};$ 
2  foreach  $s \in \varphi$  do
3      if isValid( $s$ ) then
4           $\varphi_a \leftarrow \{s\} \cup \varphi_a;$ 
5      else
6           $\varphi_s \leftarrow M(s) \cup \varphi_s;$ 
7      end
8  end
9  foreach  $s \in \varphi_s$  do
10     if isValid( $s$ ) then
11          $\varphi_a \leftarrow \{s\} \cup \varphi_a;$ 
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   $\varphi_o \leftarrow \text{infer}(\mathcal{L}, R, \Sigma); \varphi_n \leftarrow \varphi_o; M \leftarrow [];$ 
2  foreach  $s_i \in \varphi_o$  do
3      foreach  $s_j \in \varphi_o \setminus \{s_i\}$  do
4          if reduce( $s_i, s_j, \Sigma$ ) then
5               $\varphi_n \leftarrow \varphi_n \setminus s_j;$ 
6               $M \leftarrow \text{appendMap}(M, s_i, s_j);$ 
7          else if reduce( $s_j, s_i, \Sigma$ ) then
8               $\varphi_n \leftarrow \varphi_n \setminus s_i;$ 
9               $M \leftarrow \text{appendMap}(M, s_j, s_i);$ 
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(φ, M):

```

1   $\varphi_a \leftarrow \{\}; \varphi_m \leftarrow \{\};$ 
2  foreach  $s \in \varphi$  do
3      if isValid( $s$ ) then
4           $\varphi_a \leftarrow \{s\} \cup \varphi_a;$ 
5      else
6           $\varphi_s \leftarrow M(s) \cup \varphi_s;$ 
7      end
8  end
9  foreach  $s \in \varphi_s$  do
10     if isValid( $s$ ) then
11          $\varphi_a \leftarrow \{s\} \cup \varphi_a;$ 
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   $\varphi_o \leftarrow \text{infer}(\mathcal{L}, R, \Sigma); \varphi_n \leftarrow \varphi_o; M \leftarrow [];$ 
2  foreach  $s_i \in \varphi_o$  do
3      foreach  $s_j \in \varphi_o \setminus \{s_i\}$  do
4          if reduce( $s_i, s_j, \Sigma$ ) then
5               $\varphi_n \leftarrow \varphi_n \setminus s_j;$ 
6               $M \leftarrow \text{appendMap}(M, s_i, s_j);$ 
7          else if reduce( $s_j, s_i, \Sigma$ ) then
8               $\varphi_n \leftarrow \varphi_n \setminus s_i;$ 
9               $M \leftarrow \text{appendMap}(M, s_j, s_i);$ 
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(φ, M):

```

1   $\varphi_a \leftarrow \{\}; \varphi_m \leftarrow \{\};$ 
2  foreach  $s \in \varphi$  do
3      if isValid( $s$ ) then
4           $\varphi_a \leftarrow \{s\} \cup \varphi_a;$ 
5      else
6           $\varphi_s \leftarrow M(s) \cup \varphi_s;$ 
7      end
8  end
9  foreach  $s \in \varphi_s$  do
10     if isValid( $s$ ) then
11          $\varphi_a \leftarrow \{s\} \cup \varphi_a;$ 
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   $\varphi_o \leftarrow \text{infer}(\mathcal{L}, R, \Sigma); \varphi_n \leftarrow \varphi_o; M \leftarrow [];$ 
2  foreach  $s_i \in \varphi_o$  do
3      foreach  $s_j \in \varphi_o \setminus \{s_i\}$  do
4          if reduce( $s_i, s_j, \Sigma$ ) then
5               $\varphi_n \leftarrow \varphi_n \setminus s_j;$ 
6               $M \leftarrow \text{appendMap}(M, s_i, s_j);$ 
7          else if reduce( $s_j, s_i, \Sigma$ ) then
8               $\varphi_n \leftarrow \varphi_n \setminus s_i;$ 
9               $M \leftarrow \text{appendMap}(M, s_j, s_i);$ 
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(φ, M):

```

1   $\varphi_a \leftarrow \{\}; \varphi_m \leftarrow \{\};$ 
2  foreach  $s \in \varphi$  do
3      if isValid( $s$ ) then
4           $\varphi_a \leftarrow \{s\} \cup \varphi_a;$ 
5      else
6           $\varphi_s \leftarrow M(s) \cup \varphi_s;$ 
7      end
8  end
9  foreach  $s \in \varphi_s$  do
10     if isValid( $s$ ) then
11          $\varphi_a \leftarrow \{s\} \cup \varphi_a;$ 
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   $\varphi_o \leftarrow \text{infer}(\mathcal{L}, R, \Sigma); \varphi_n \leftarrow \varphi_o; M \leftarrow [];$ 
2  foreach  $s_i \in \varphi_o$  do
3      foreach  $s_j \in \varphi_o \setminus \{s_i\}$  do
4          if reduce( $s_i, s_j, \Sigma$ ) then
5               $\varphi_n \leftarrow \varphi_n \setminus s_j;$ 
6               $M \leftarrow \text{appendMap}(M, s_i, s_j);$ 
7          else if reduce( $s_j, s_i, \Sigma$ ) then
8               $\varphi_n \leftarrow \varphi_n \setminus s_i;$ 
9               $M \leftarrow \text{appendMap}(M, s_j, s_i);$ 
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(φ, M):

```

1   $\varphi_a \leftarrow \{\}; \varphi_m \leftarrow \{\};$ 
2  foreach  $s \in \varphi$  do
3      if isValid( $s$ ) then
4           $\varphi_a \leftarrow \{s\} \cup \varphi_a;$ 
5      else
6           $\varphi_s \leftarrow M(s) \cup \varphi_s;$ 
7      end
8  end
9  foreach  $s \in \varphi_s$  do
10     if isValid( $s$ ) then
11          $\varphi_a \leftarrow \{s\} \cup \varphi_a;$ 
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.



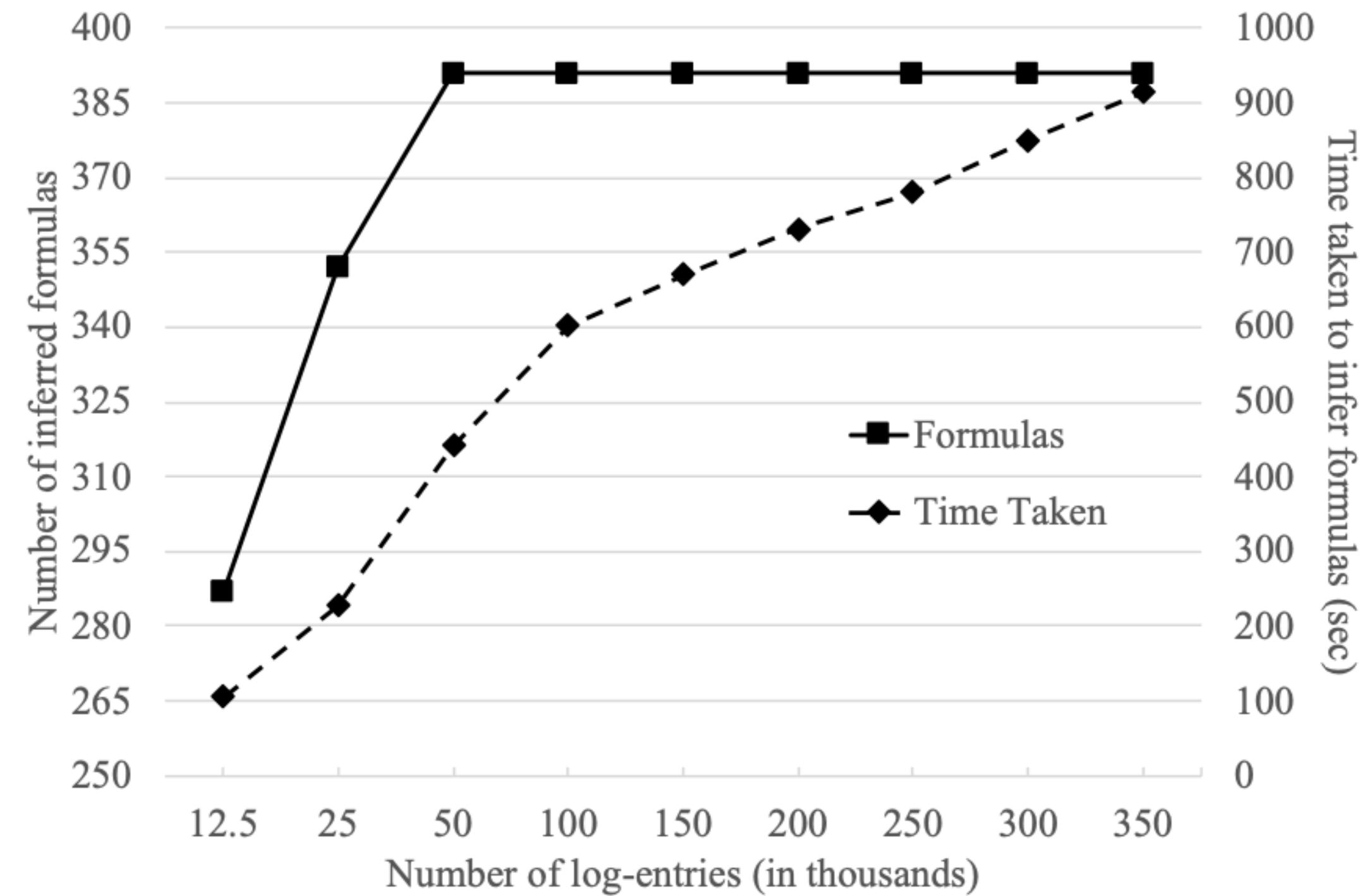
Prototype Evaluation

- 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

