

ProProv

ProProv

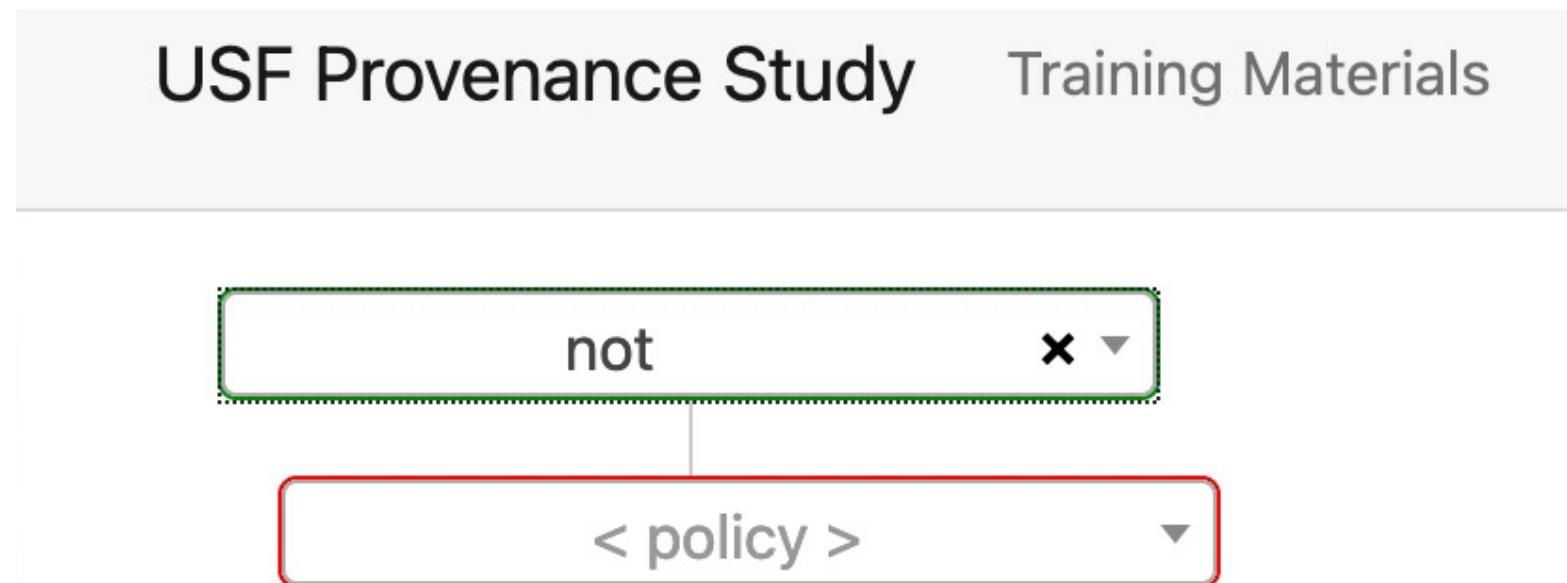
- Language and system for specifying provenance policies over graphs representing the provenance of secure computations
- Policies are expressed as logical statements, or predicates, over provenance graphs
- Contains a graphical user interface to specify and test policies

ProProv Syntax

- Provenance node types t : accountAgent, nodeAgent, agent, dataEntity, contractEntity, keyEntity, entity, or activity
- Edges e : wasDerivedFrom, wasAttributedTo, wasGeneratedBy, used, actedOnBehalfOf, or wasAssociatedWith
- Variables $x \in Strings$
- Nodes $n \in Strings$
- Policies p : $!p$, $p_1 \wedge p_2$, $p_1 \vee p_2$, $\text{forall } x: t. p$, $\text{exists } x: t. p$, $p_1 \Rightarrow p_2$, $e(n_1, n_2)$

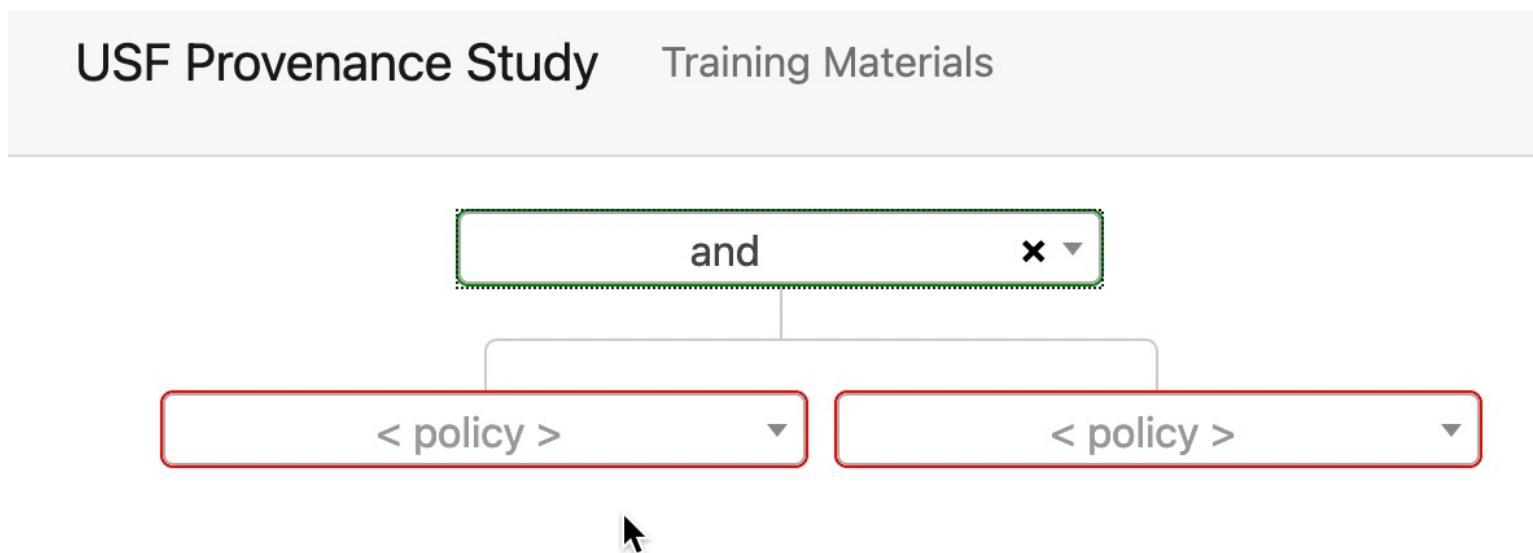
Negation policy ($\neg p$)

- Asserts that policy p is not true



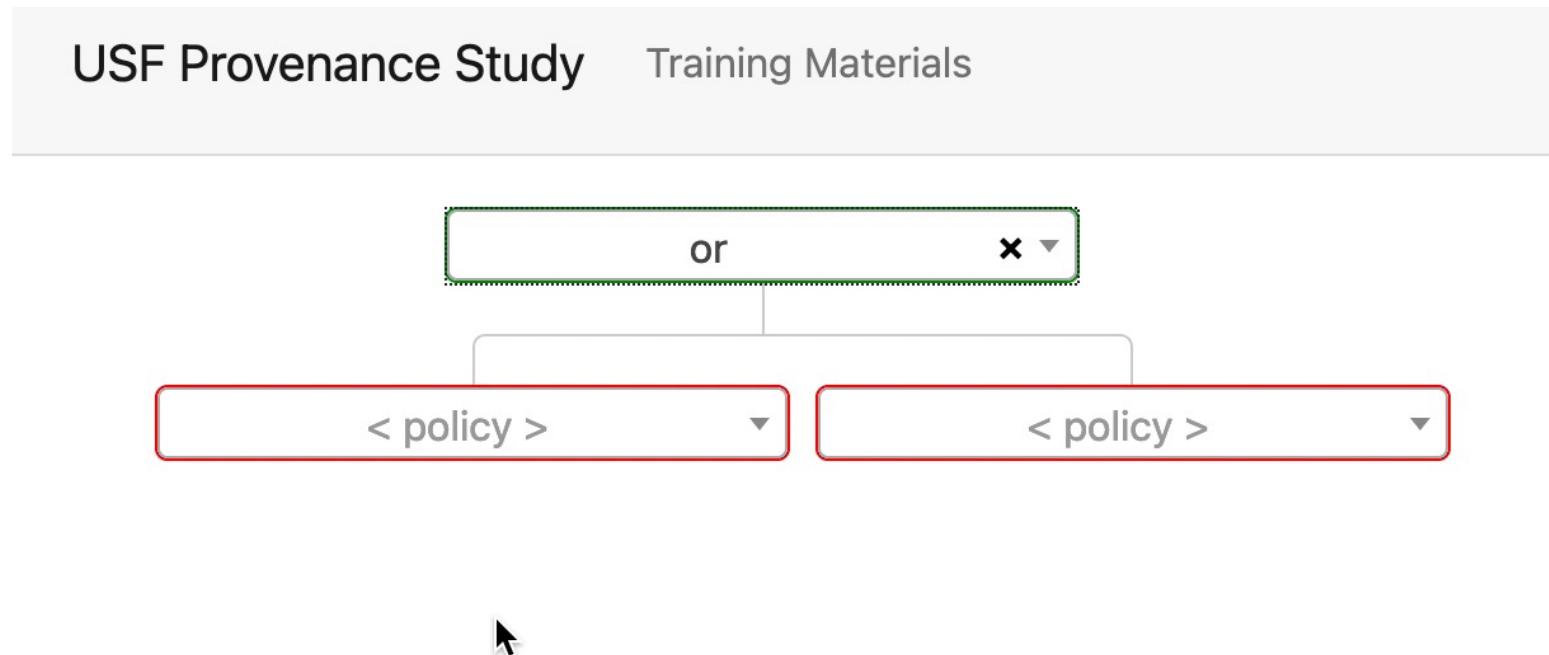
Conjunction Policy ($p_1 \wedge p_2$)

- Also called AND
- Asserts that both policy p_1 and policy p_2 are true



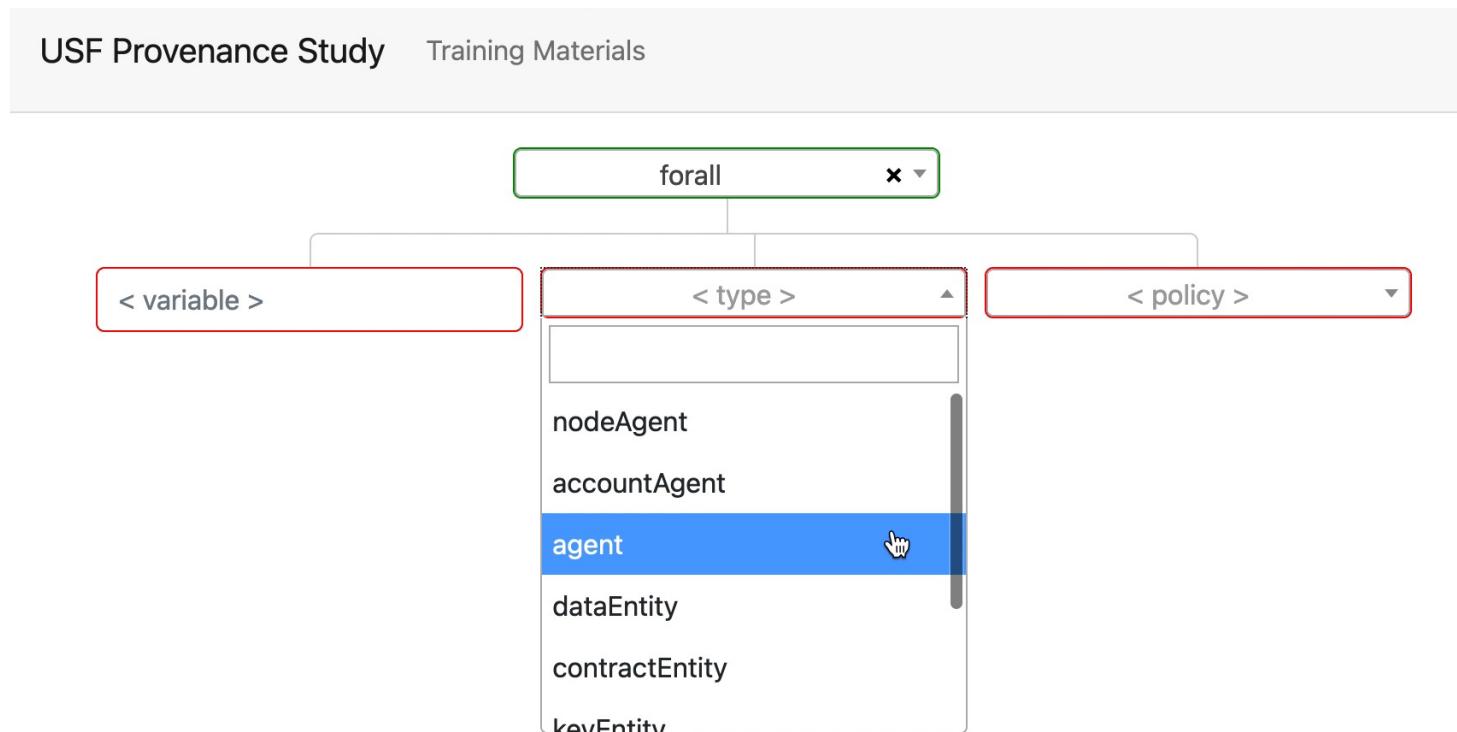
Disjunction Policy ($p_1 \vee p_2$)

- Also called OR
- Asserts that either policy p_1 is true, policy p_2 is true , or both p_1 and p_2 are true



Universal Policy ($\forall x: t. p$)

- Asserts that policy p is true for every provenance node of type t . Variable x can be used in p



Existential Policy ($\exists x: t. p$)

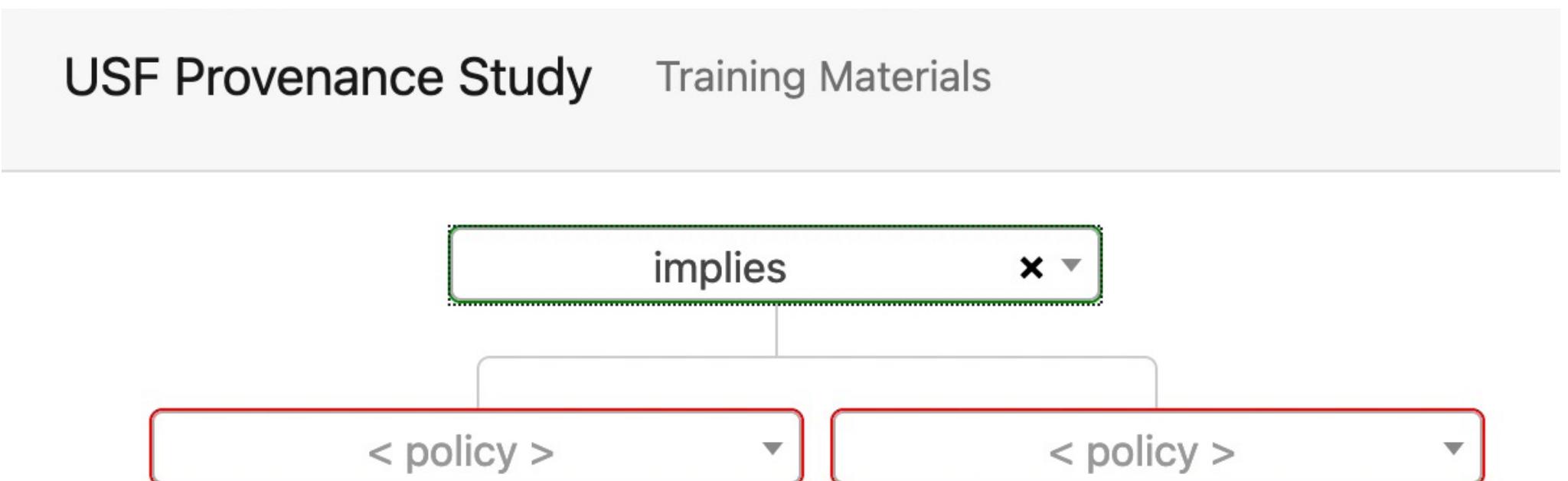
- Asserts that policy p is true for some provenance node of type t . Variable x can be used in p

USF Provenance Study Training Materials

The screenshot shows a user interface for defining a policy. At the top, there's a green button labeled "exists" with a close icon. Below it, three input fields are outlined in red: "< variable >", "< type >", and "< policy >". The "< type >" field is currently active, displaying a dropdown menu with several options: "nodeAgent" (highlighted in blue), "accountAgent", "agent", "dataEntity", "contractEntity", and "keyEntity". A cursor icon is visible next to "accountAgent".

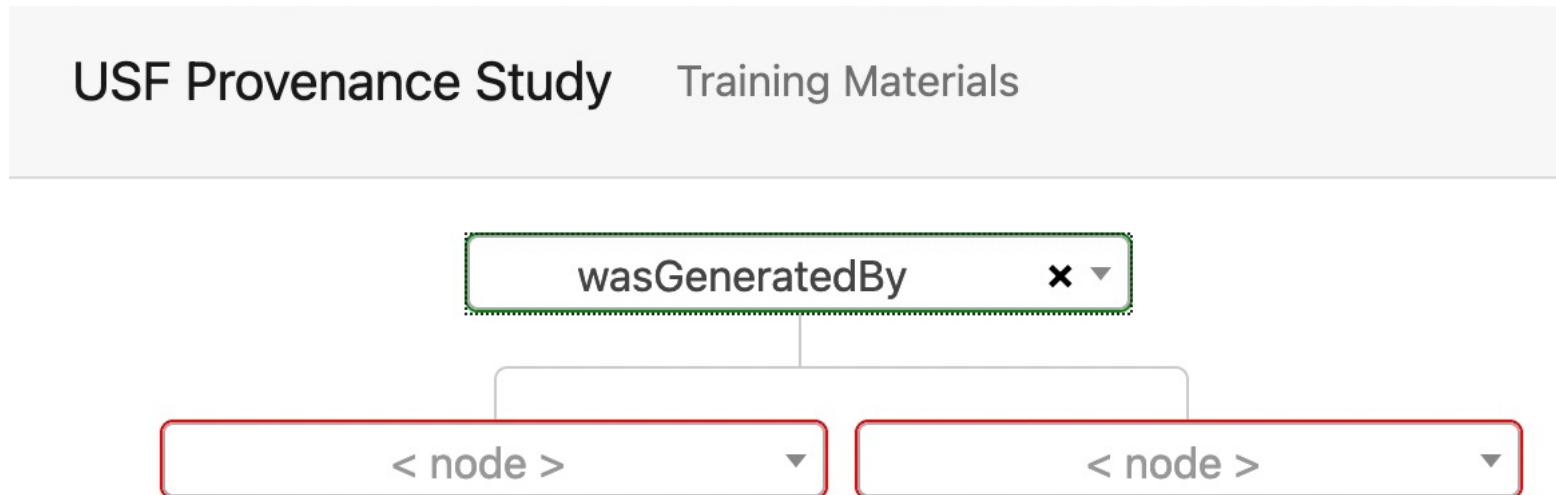
Implication policy ($p_1 \Rightarrow p_2$)

- Asserts that if policy p_1 is true then policy p_2 is true



Edge policy ($e(n_1, n_2)$)

- e : wasAttributedTo, wasDerivedFrom, wasGeneratedBy, used, actedOnBehalfOf, or wasAssociatedWith
- Asserts that there is an edge with label e between node n_1 and node n_2



Provenance policy examples

- ProProv is case insensitive, $\text{Average} \equiv average$
- Write a policy to ensure that activity Average used data entity SalaryB

USF Provenance Study Training Materials

Save and Continue S00001

used

Average SalaryB

Write a policy to ensure that activity Average used data entity SalaryB

Input 1 Input 2 Input 3 Input 4 Input 5

1

✓

SalaryB → U → Average

U: Used

Evaluate used(Average, SalaryB)

```
graph LR; SalaryB((SalaryB)) -- U --> Average((Average))
```

Provenance policy examples

- Write a policy to ensure that data entity *AverageSalary* was not derived from *SalaryC*

USF Provenance Study Training Materials Save and Continue S00001

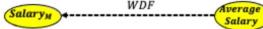
Write a policy to ensure that data entity AverageSalary was not derived from SalaryC

Input 1 Input 2 Input 3 Input 4 Input 5

1

`not
wasDerivedFrom
AverageSalary x SalaryC x`




WDF:WasDerivedFrom

Evaluate !wasDerivedFrom(AverageSalary, SalaryC)

Provenance policy examples

- Write a policy to ensure that activity *Average* used key entity *KeyB* and *KeyB* was attributed to account agent *Bob*

USF Provenance Study Training Materials Save and Continue S00001

and

used

wasAttributedTo

Average

KeyB

KeyB

Bob

Write a policy to ensure that activity Average used key entity KeyB and KeyB was attributed to account agent Bob

Input 1 Input 2 Input 3 Input 4 Input 5

1

✓

Bob — WAT — KeyB — U — Average

U: Used
WAT: WasAttributedTo

Evaluate used(Average, KeyB) \wedge wasAttributedTo(KeyB, Bob)

The screenshot shows a web-based interface for defining provenance policies. On the left, there's a tree-like structure for building a query. The root node is 'and'. It has two children: 'used' and 'wasAttributedTo'. The 'used' node has two leaf nodes: 'Average' and 'KeyB'. The 'wasAttributedTo' node also has two leaf nodes: 'KeyB' and 'Bob'. To the right of this tree is a text input field containing the policy statement: "Write a policy to ensure that activity Average used key entity KeyB and KeyB was attributed to account agent Bob". Below this text input are five tabs labeled 'Input 1' through 'Input 5', with 'Input 1' currently selected. A large green checkmark icon is positioned to the right of the tabs. At the bottom of the interface, there's a diagram illustrating the provenance graph. It consists of four nodes: 'Bob' (orange trapezoid), 'KeyB' (yellow oval), 'Average' (blue rectangle), and another 'KeyB' node (yellow oval). Directed edges connect them: an edge from 'Bob' to the first 'KeyB' labeled 'WAT', an edge from the first 'KeyB' to the 'Average' node labeled 'U', and an edge from the second 'KeyB' back to the 'Average' node labeled 'U'. Below the diagram is a legend box containing the labels 'U: Used' and 'WAT: WasAttributedTo'. At the very bottom of the interface is a button labeled 'Evaluate' followed by the expression 'used(Average, KeyB) \wedge wasAttributedTo(KeyB, Bob)'.

Provenance policy examples

- Write a policy to ensure that there is some data entity that *AverageSalary* was derived from

USF Provenance Study Training Materials

exists \exists

d dataEntity $\times \downarrow$

wasDerivedFrom $\times \downarrow$

AverageSalary $\times \downarrow$

d $\times \downarrow$

Save and Continue S00001

Write a policy to ensure that there is some data entity that AverageSalary was derived from

Input 1 Input 2 Input 3 Input 4 Input 5

1

✓

WDF:WasDerivedFrom

Evaluate exists d:dataEntity. wasDerivedFrom(AverageSalary, d)

```
graph LR; Salaryc -- "WDF" --> AverageSalary; Salaryg -- "WDF" --> AverageSalary;
```

Provenance policy examples

- Write a policy to ensure for every data entity, if activity *Average* used the data entity, then the data entity was attributed to either Bob, Alice, or Mallory



ProProv Incorrect Answers

USF Provenance Study Training Materials

Save and Continue S00001

wasAttributedTo

KeyA Bob

Write a policy to ensure for every data entity, if activity Average used the data entity, then the data entity was attributed to either Bob, Alice, or Mallory

Input 1 Input 2 Input 3 Input 4 Input 5

1

Unexpected result: Graph 1 should satisfy your policy.
Unexpected result: Graph 2 should satisfy your policy.
Unexpected result: Graph 4 should satisfy your policy.

OK

Bob \xleftarrow{WAT} Salary_B \xrightarrow{U} Average

U: Used
WAT: WasAttributedTo

Evaluate wasAttributedTo(KeyA, Bob)