Determining the Cause of a Design Model Inconsistency

Lars Van Roy
dept. of Mathematics and Computer Science
University of Antwerp
lars.vanroy@student.uantwerpen.be

January 19, 2021

Overview

1. Paper overview
2. Problem overview
3. Approach
    3.1 Generate the validation tree
    3.2 Calculate the cause
4. Conclusion

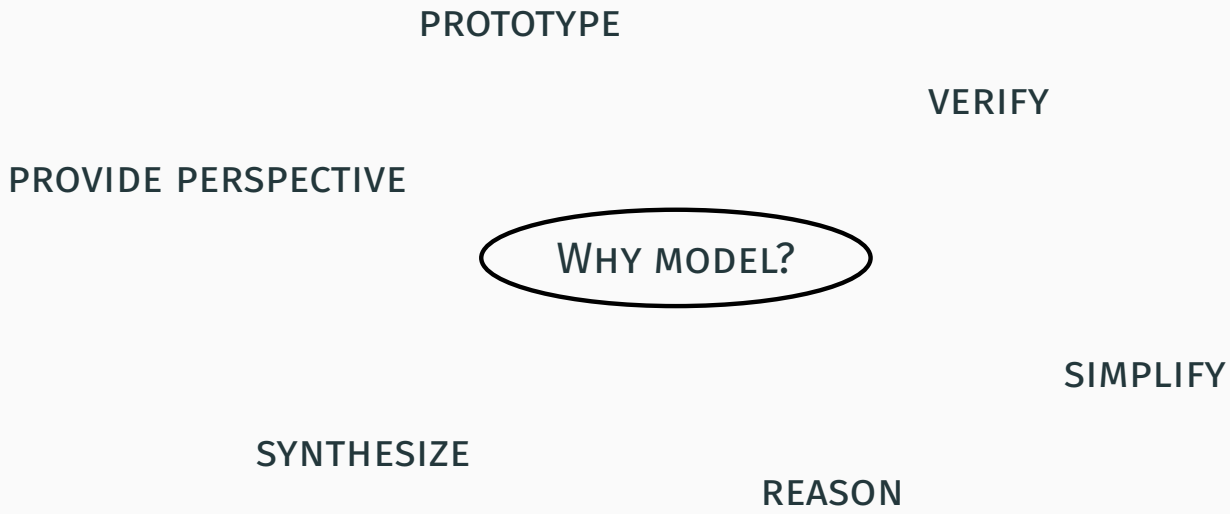
Paper overview

Paper overview

- Writen by
  - Alexander Reder
  - Alexander Egyed
- 2013
- IEEE Transactions on Software Engineering

Problem overview

Problem overview I



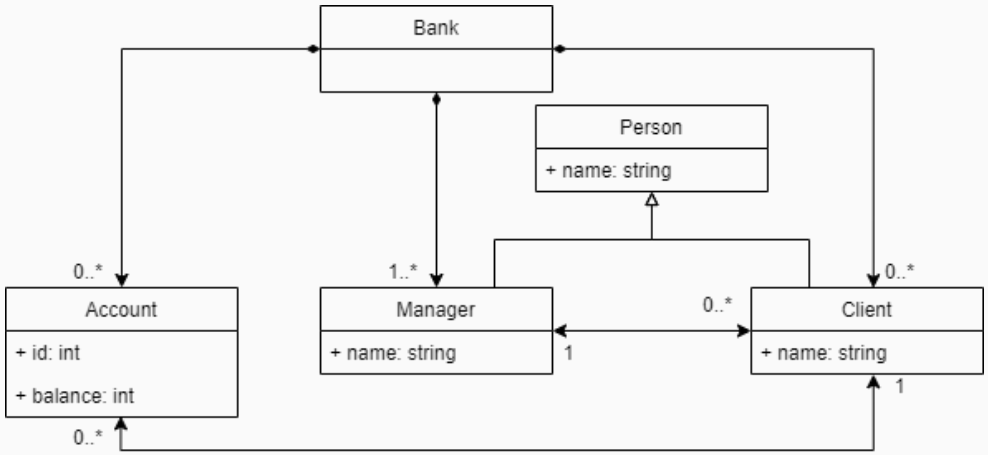
Problem overview II

- Only works if consistent with constraints
- Need to resolve any inconsistencies

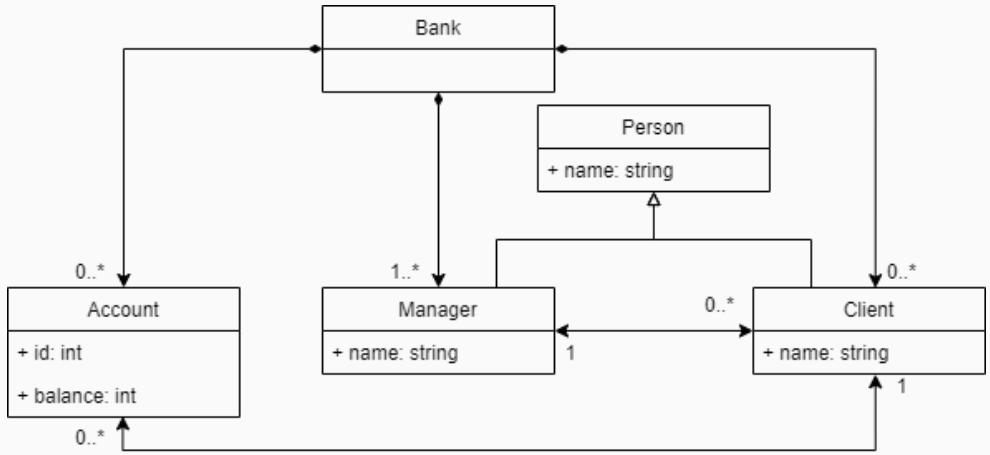
Problem overview II

- Only works if consistent with constraints
- Need to resolve any inconsistencies
  - Which constraints are violated
  - What causes these violations

Problem overview III



Problem overview III



Attribute names must be unique within their class hierarchy

Approach

Approach I

- Meta level
- Two step approach

Approach I

- Meta level
- Two step approach
    1. Generate the validation tree

Approach I

- Meta level
- Two step approach
  1. Generate the validation tree
  2. Look for potential cause of violation

Generate validation tree

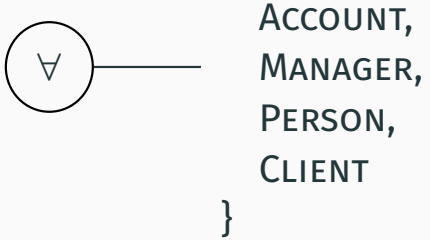
- Convert the execution of the constraint to a tree
- Nodes are expressions
- Leaves are model defined values

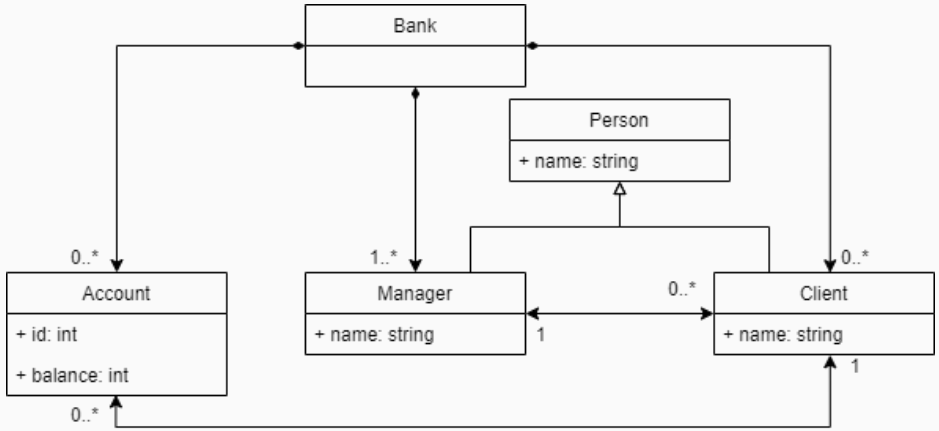
Generate validation tree II

```
foreach class do
    let c_attrs ← class.attributes
    foreach parent in class.parents do
        let p_attrs ← parent.attributes
        foreach attr in c_attrs do
            attr not in p_attrs
        end
    end
end
```

Generate validation tree III

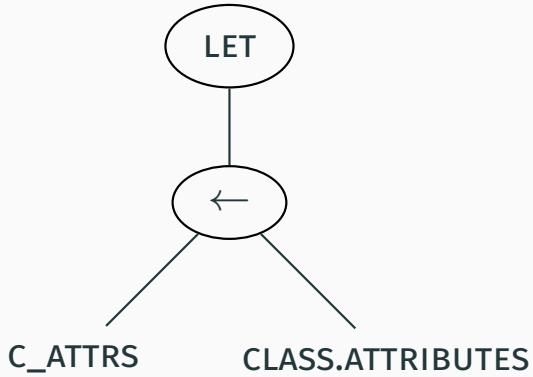
```
foreach class do
    let c_attrs ← class.attributes
    foreach parent in class.parents do
        let p_attrs ← parent.attributes
        foreach attr in c_attrs do
            attr not in p_attrs
        end
    end
end
```

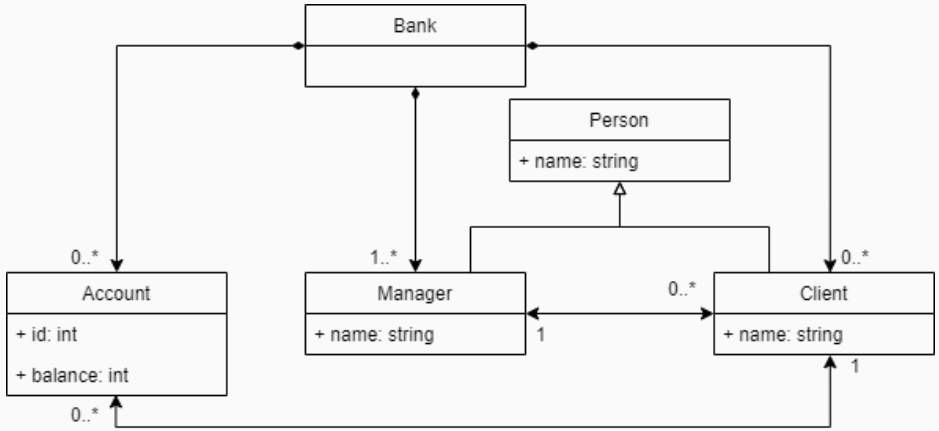




Generate validation tree IV

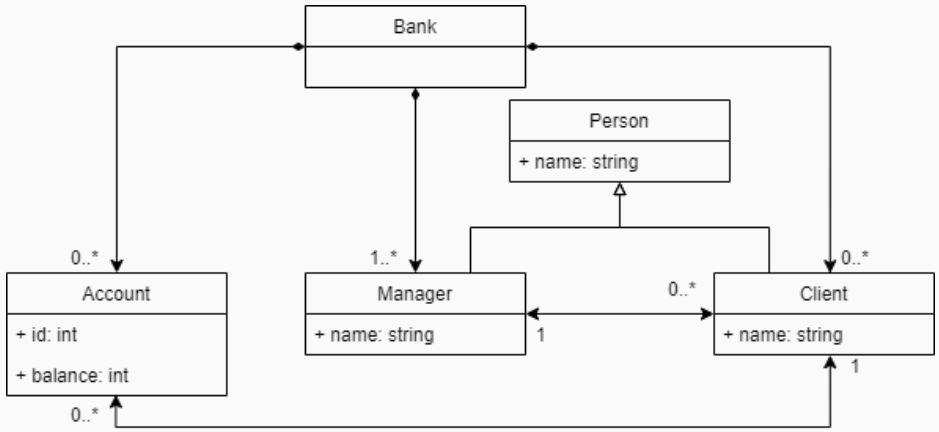
```
foreach class do
    let c_attrs ← class.attributes
    foreach parent in class.parents do
        let p_attrs ← parent.attributes
        foreach attr in c_attrs do
            attr not in p_attrs
        end
    end
end
```





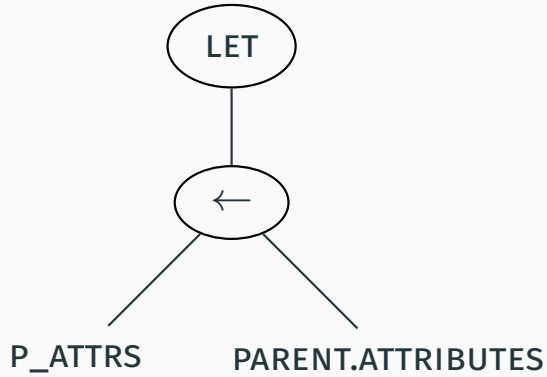
Generate validation tree V

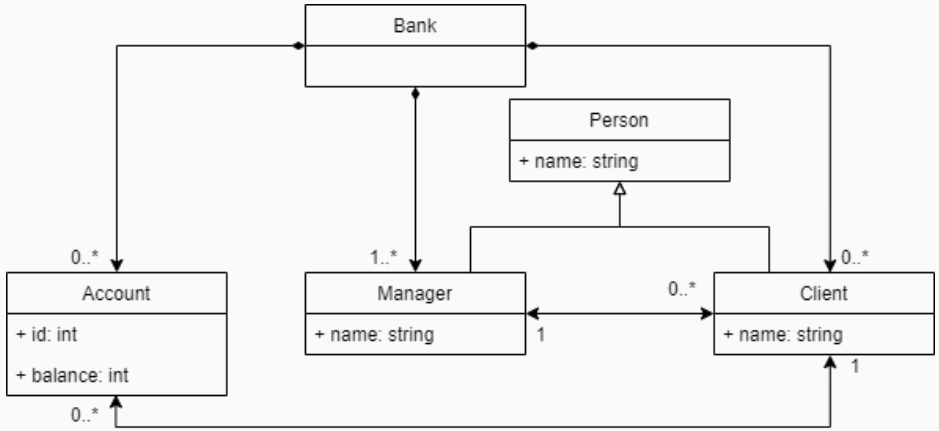
```
foreach class do
    let c_attrs ← class.attributes
    foreach parent in class.parents do
        let p_attrs ← parent.attributes
        foreach attr in c_attrs do
            attr not in p_attrs
        end
    end
end
```



Generate validation tree VI

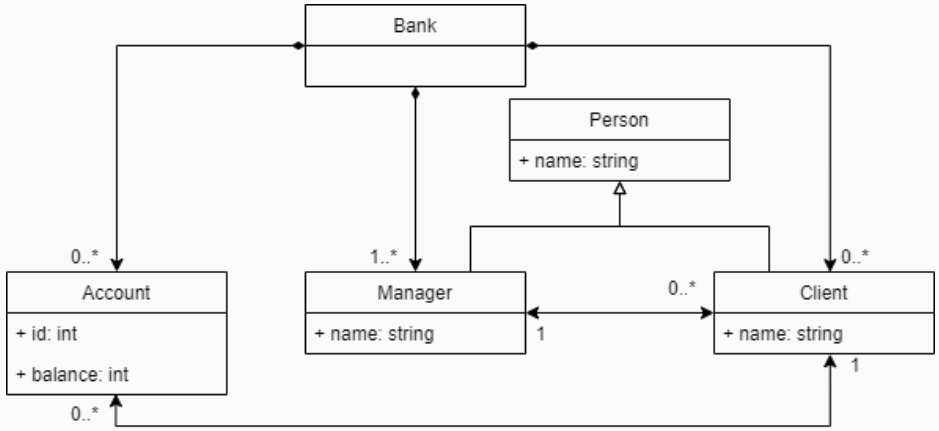
```
foreach class do
    let c_attrs ← class.attributes
    foreach parent in class.parents do
        let p_attrs ← parent.attributes
        foreach attr in c_attrs do
            attr not in p_attrs
        end
    end
end
```





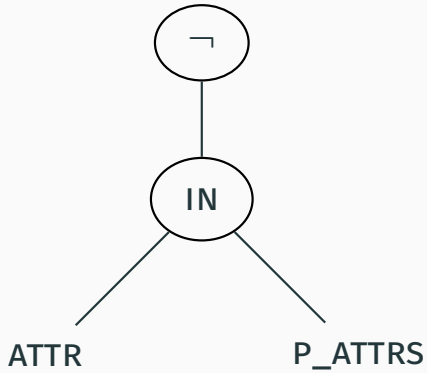
Generate validation tree VII

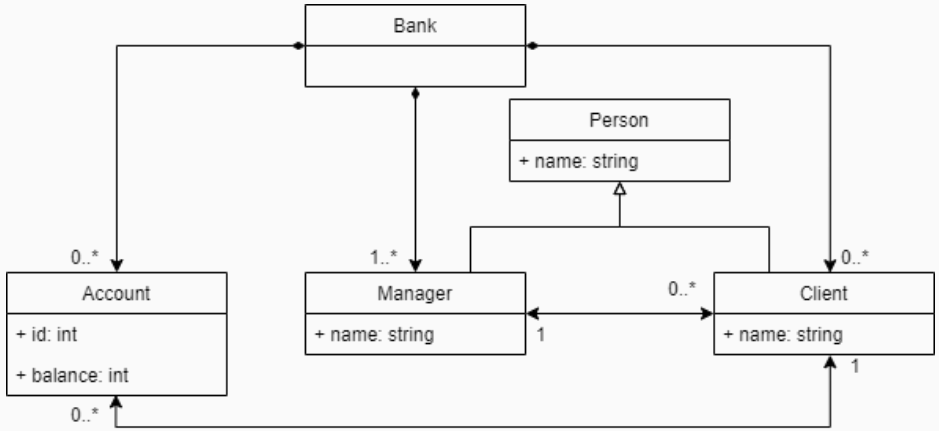
```
foreach class do
    let c_attrs ← class.attributes
    foreach parent in class.parents do
        let p_attrs ← parent.attributes
        foreach attr in c_attrs do
            attr not in p_attrs
        end
    end
end
```



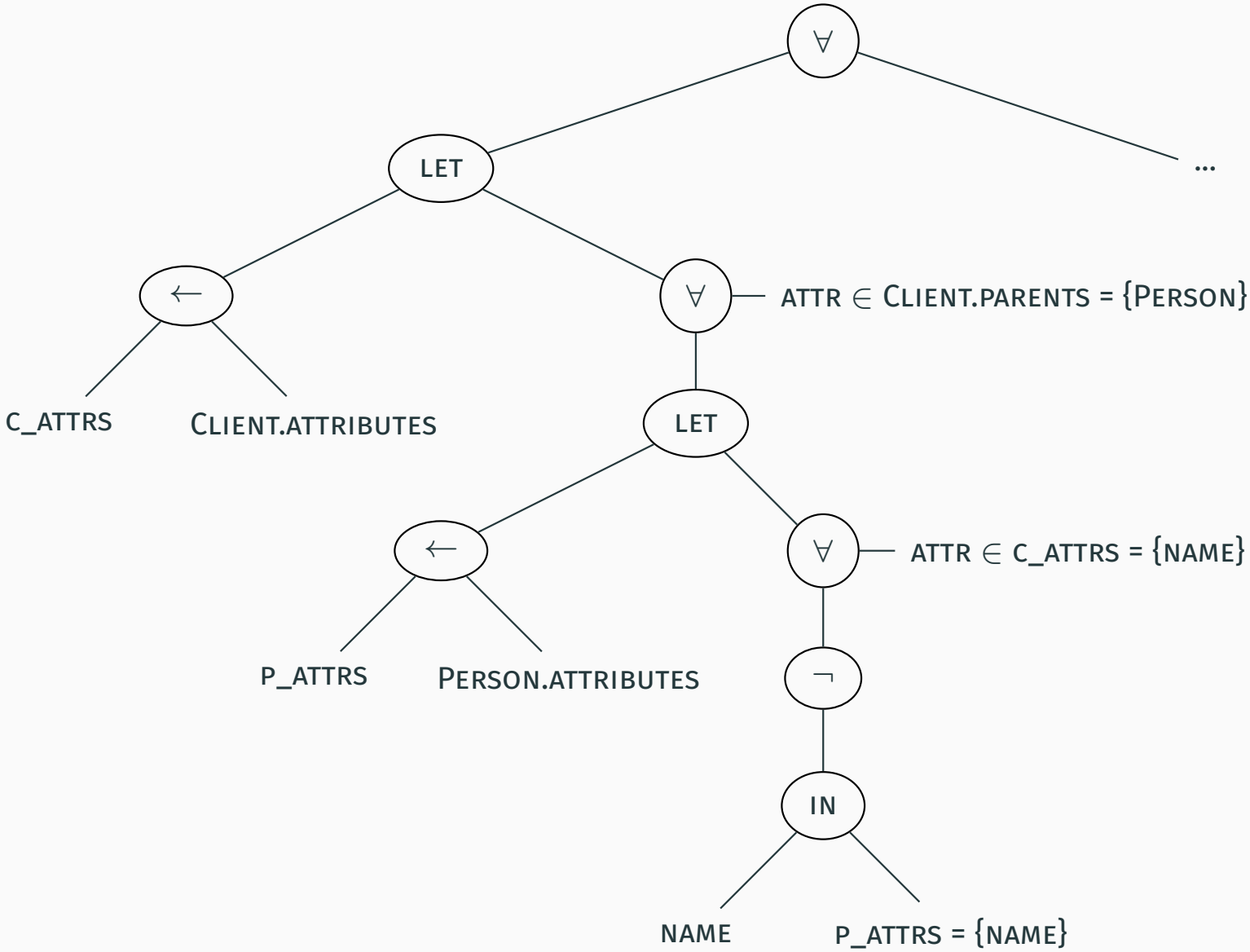
Generate validation tree VIII

```
foreach class do
    let c_attrs ← class.attributes
    foreach parent in class.parents do
        let p_attrs ← parent.attributes
        foreach attr in c_attrs do
            attr not in p_attrs
        end
    end
end
```

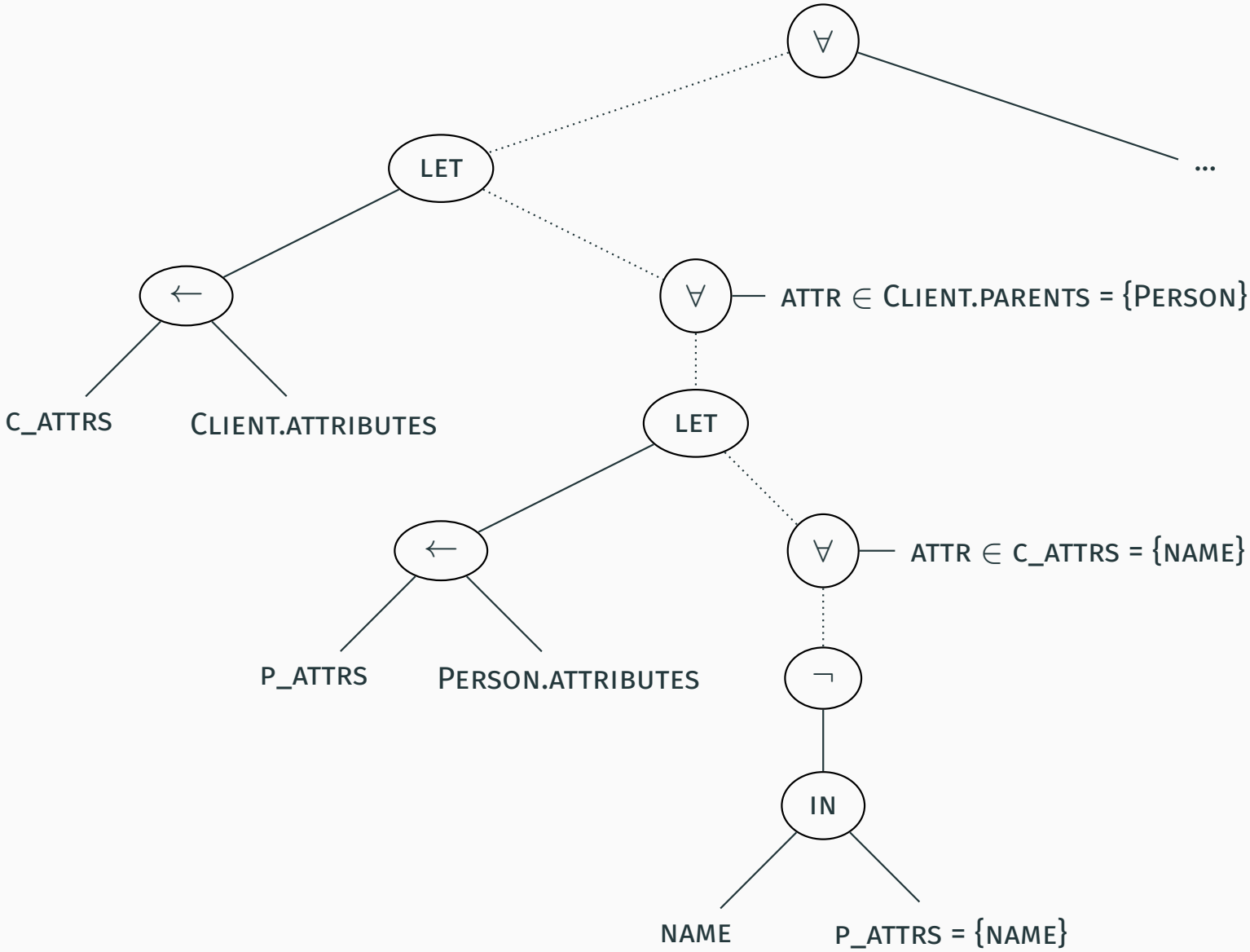




Generate validation tree IX



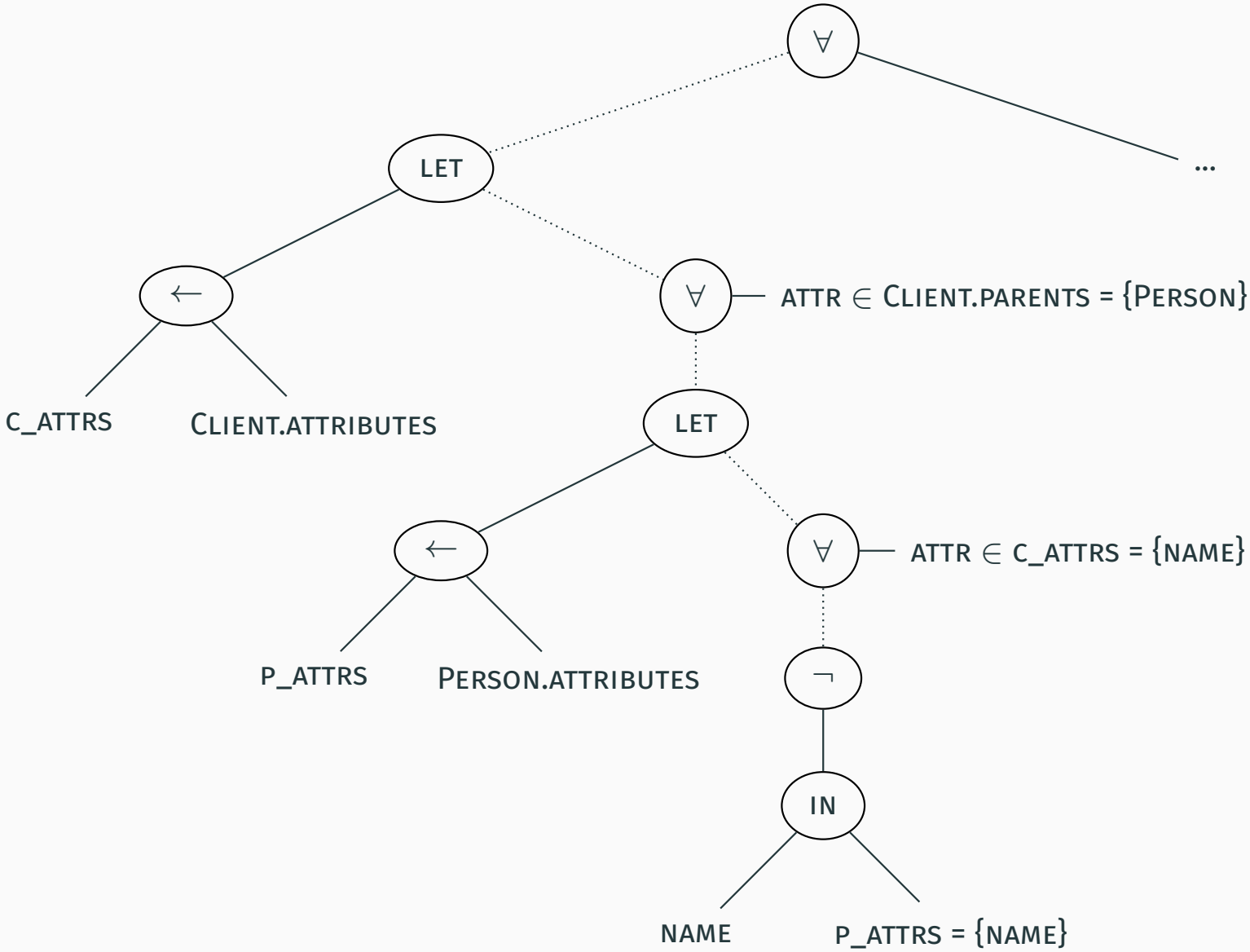
Generate validation tree IX



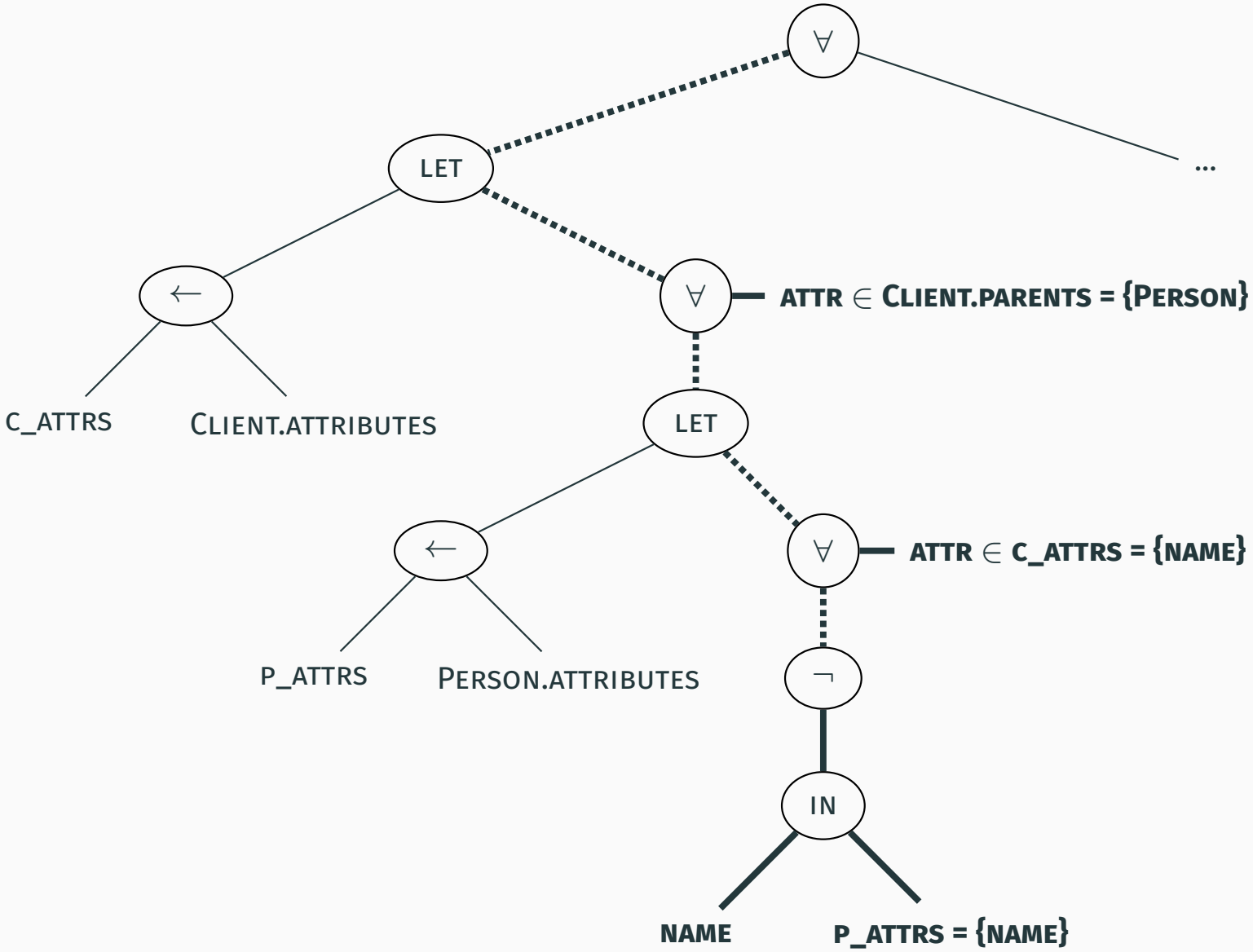
Cause of violation I

- We know that there is a violation
- What caused it
- Top down traversal of the tree
- Find all items that could have contributed
  - Expressions
  - Model defined values

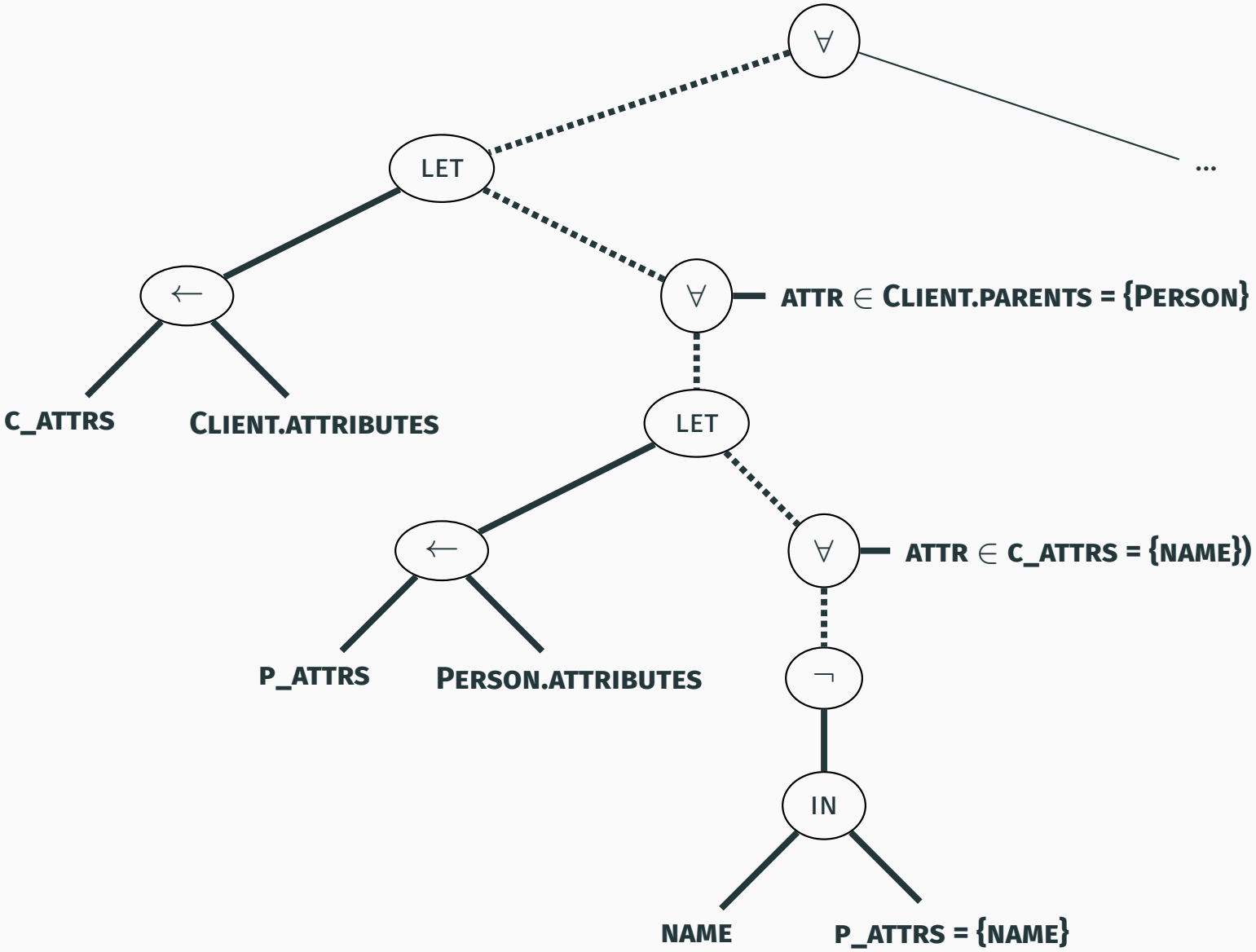
Cause of violation II



Cause of violation II

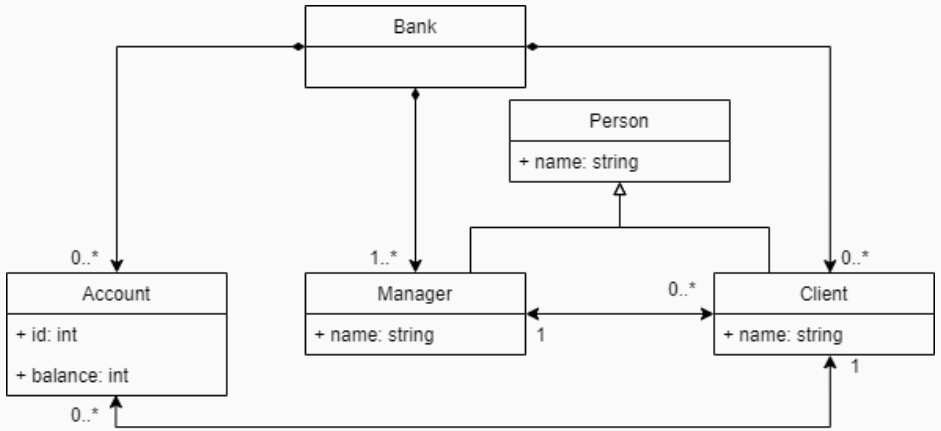


Cause of violation II



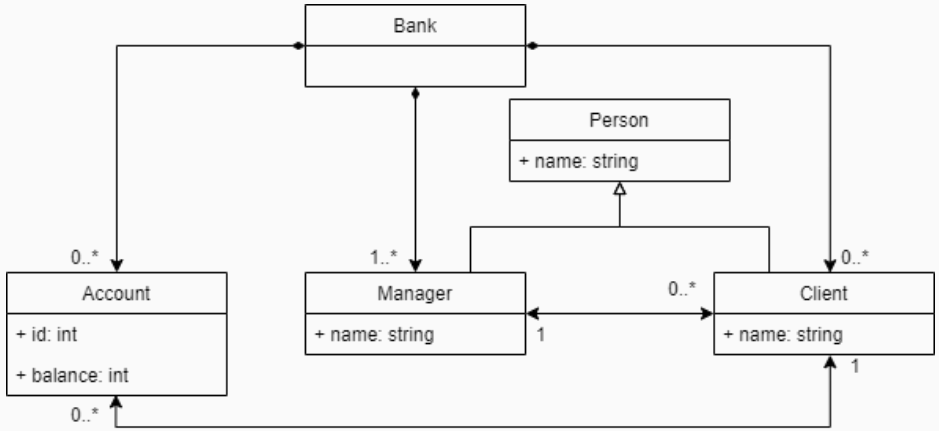
Cause of violation III

- We now know which elements could have contributed to the violation
  - The classes Manager, Client and Person
  - The parent attribute of Manager and Client
  - The name attribute of Manager, Client and Person
  - The attributes of Manager, Client and Person



Cause of violation III

- We now know which elements could have contributed to the violation
  - The classes Manager, Client and Person
  - The parent attribute of Manager and Client
  - The name attribute of Manager, Client and Person
  - The attributes of Manager, Client and Person
- Previous research would have only given the name attribute



Conclusion

Conclusion

- Modelling only worthwhile if consistent with constraints

Conclusion

- Modelling only worthwhile if consistent with constraints
- There is a need for proper cause of inconsistency

Conclusion

- Modelling only worthwhile if consistent with constraints
- There is a need for proper cause of inconsistency
- The proposed approach gives an efficient way of providing
  - model elements
  - expressions part of the constraint

Conclusion

- Modelling only worthwhile if consistent with constraints
- There is a need for proper cause of inconsistency
- The proposed approach gives an efficient way of providing
  - model elements
  - expressions part of the constraint
- still need for means to actually resolve inconsistency