

# QFLan: A tool for the Quantitative Analysis of Highly Reconfigurable Systems

Andrea Vandin

Sant'Anna School of Advanced Studies Pisa, Italy

DTU Technical University of Denmark

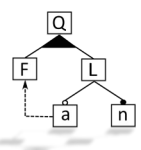
Maurice H. ter Beek  
ISTI CNR Pisa, Italy

Axel Legay  
UCLouvain, Belgium

Alberto Lluch Lafuente  
DTU, Denmark

Classes 21t-22t, Software Validation and Verification, Unipi, 04-05/12/2023

Class 21t 04/12/2023



## References

**[JSS22]** Roberto Casaluze, Andrea Burattin, Francesca Chiaromonte, Alberto Lluch Lafuente, Andrea Vandin, White-box validation of quantitative product lines by statistical model checking and process mining [Minor revision]

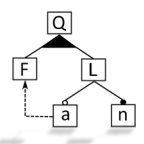
**[TSE18]** Maurice ter Beek, Axel Legay, Alberto Lluch Lafuente, Andrea Vandin, A framework for quantitative modeling and analysis of highly (re)configurable systems, IEEE Transactions on Software Engineering (TSE), 2018.

**[FM18]** Andrea Vandin, Maurice ter Beek, Axel Legay, Alberto Lluch Lafuente, QFLan: A Tool for the Quantitative Analysis of Highly Reconfigurable Systems.

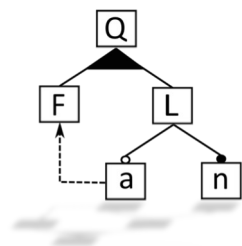
**[ISOLA16]** Maurice ter Beek, Axel Legay, Alberto Lluch Lafuente, Andrea Vandin, Statistical Model Checking for Product Lines.

**[SPLC15]** Maurice ter Beek, Axel Legay, Alberto Lluch Lafuente, Andrea Vandin, Statistical Analysis of Probabilistic Models of Software Product Lines with Quantitative Constraints.

**[FMSPLE15]** Maurice ter Beek, Axel Legay, Alberto Lluch Lafuente, Andrea Vandin, Quantitative Analysis of Probabilistic Models of Software Product Lines with Statistical Model Checking.



Presented in [FM'18][TSE'18]  
 Prototypes in [FMSPL'15][SPLC'15][ISOLA'16]

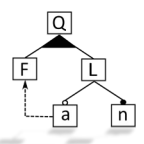


The screenshot displays the QFLan IDE interface for a VendingMachine model. The interface is divided into several panes:

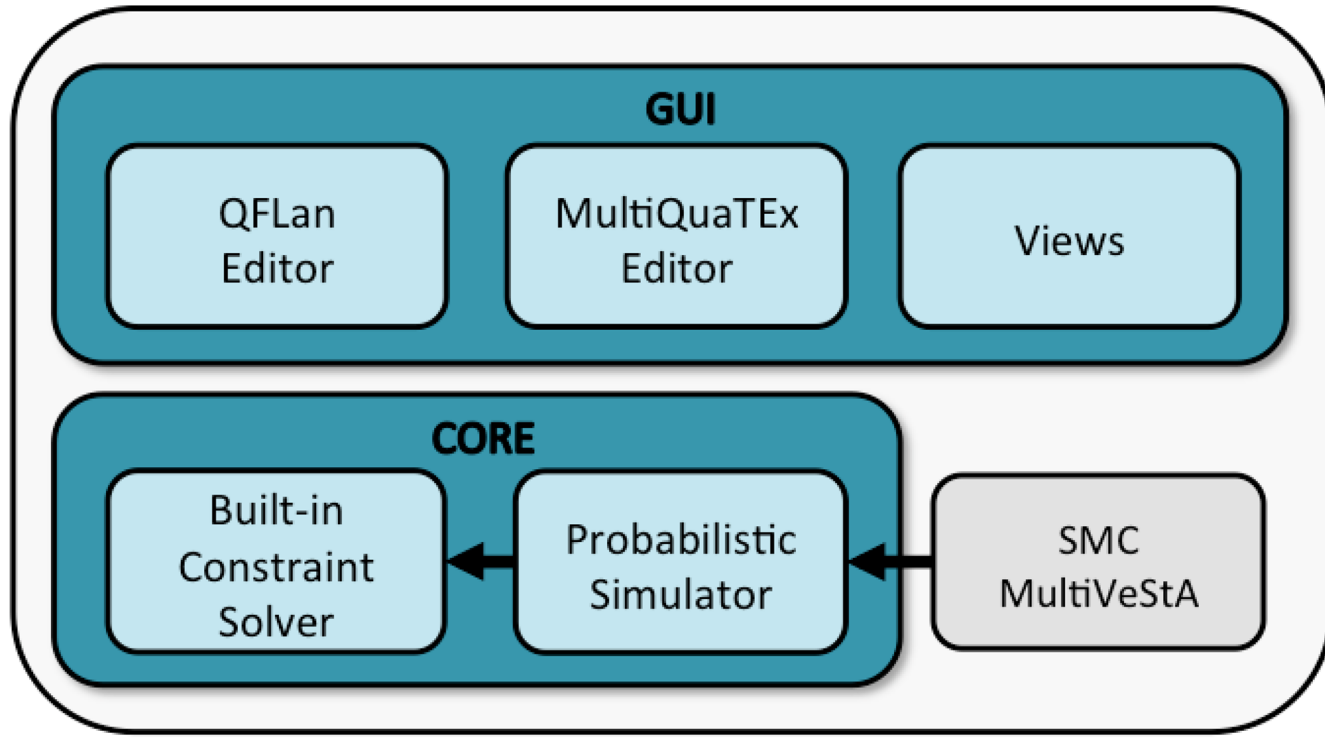
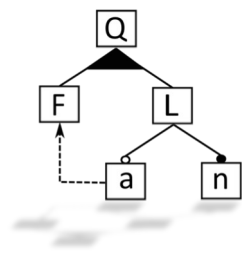
- Project Explorer:** Shows the project structure, including benchmarks, fse2018, MultiVeStA\_OUTPUT, src-gen, VendingMachine.qflan, OtherQFLanProject, and QFLAN\_Examples.
- QFLan Editor:** Contains the model definition code:
 

```

begin model VendingMachine
  begin variables
  begin abstract features
    Machine Beverage CoffeeBased
  end abstract features
  begin concrete features
    Cocoa Tea Cappuccino Coffee
  end concrete features
  begin feature diagram
    Machine -> {?Cocoa, Beverage}
    Beverage -XOR-> {CoffeeBased, Tea}
    CoffeeBased -OR-> {Cappuccino, Coffee}
  end feature diagram
end model
      
```
- Outline View:** Shows a hierarchical view of the model's components, including 2 variables, 3 abstract features, 4 concrete features (Cocoa, Tea, Cappuccino, Coffee), 3 feature relations, 2 cross tree constraints, 1 feature predicate, 1 quantitative constraint, 8 actions, 1 action constraint, and 1 process.
- Console View:** Displays the execution log, showing MultiVeStA client progress and simulation results across iterations 23, 24, 25, and 26.
- Plot View:** A line graph titled "MultiVeStA analysis of VendingMachine.qflan" showing "Means estimations" on the y-axis (ranging from -0.93 to 10.22) against "x" on the x-axis (ranging from -1 to 101). The plot shows five data series: obs1AtStep(x) (blue), obs2AtStep(x) (red), obs3AtStep(x) (green), obs4AtStep(x) (black), and obs5AtStep(x) (purple). The blue series shows a sharp peak around x=5, while the other series remain relatively flat near zero.

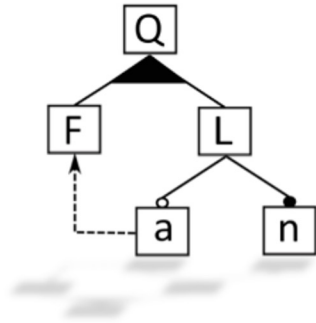


Presented in [FM'18][TSE'18]  
Prototypes in [FMSPLE'15][SPLC'15][ISOLA'16]



# QFLan

*A framework for quantitative modeling and analysis of highly (re)configurable systems*



## Summary

*QFLan* is a software tool for the modeling and analysis of highly reconfigurable systems, including software product lines.

The tool offers an easy-to-use, rule-based probabilistic language to specify models with probabilistic behaviour. Quantitative constraints can be used to restrict the class of admissible configurations (or products), like (using a family of reconfigurable vending machines from [here](#)):

- machines can have a certain maximum cost,
- machines serving coffee-based beverages cannot sell tea,
- in order to serve cappuccino it is necessary to have the feature of serving also coffee,

Also it is possible to express conditions like:

- machines serving cappuccino provided with a coca dispenser can serve chocaccino.

*QFLan* has been combined with the distributed statistical model checker [MultiVeStA](#) to perform

▼ Pages 6

[Home](#)

[Install QFLan](#)

[Models from FM18 paper](#)

[Models from TSE18 paper](#)

[Publications](#)

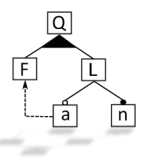
[Source code](#)

+ Add a custom sidebar

Clone this wiki locally

<https://github.com/qflan/qflan>

Clone in Desktop



## Feature Model

- Abstract and Concrete Features
- Cross-tree Constraints
- Quantitative Constraints

## Behaviour

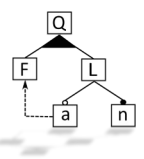
- Actions and Action Constraints
- Transitions
- Initial Configuration

## MultiVeStA Analysis

- Analysis when a condition holds
- Analysis at varying of time

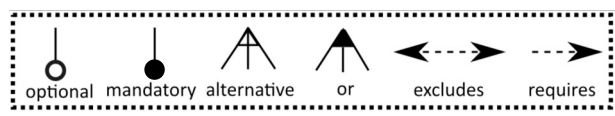
## An Application to a Simple Security Scenario

- Schneier's SafeLock Attack Tree

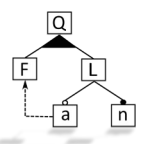


# A simple vending machine product line

The feature model

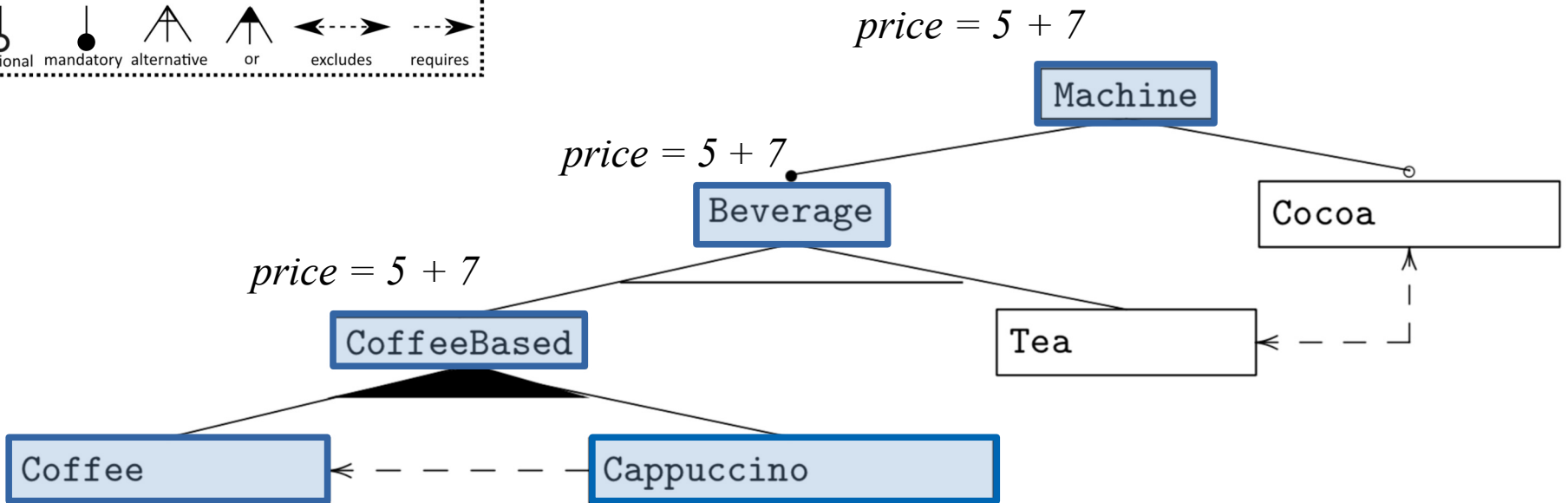
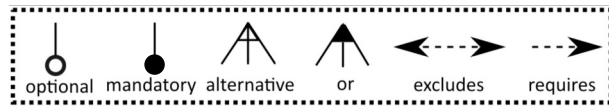


Machine



# A simple vending machine product line

The feature model: Abstract & Concrete Features



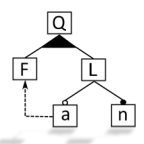
```
begin abstract features
Machine Beverage CoffeeBased
end abstract features
```

```
begin concrete features
Cocoa Tea Cappuccino Coffee
end concrete features
```

```
begin feature diagram
Machine -> {?Cocoa, Beverage}
Beverage -XOR-> {CoffeeBased, Tea}
CoffeeBased -OR-> {Cappuccino, Coffee}
end feature diagram
```

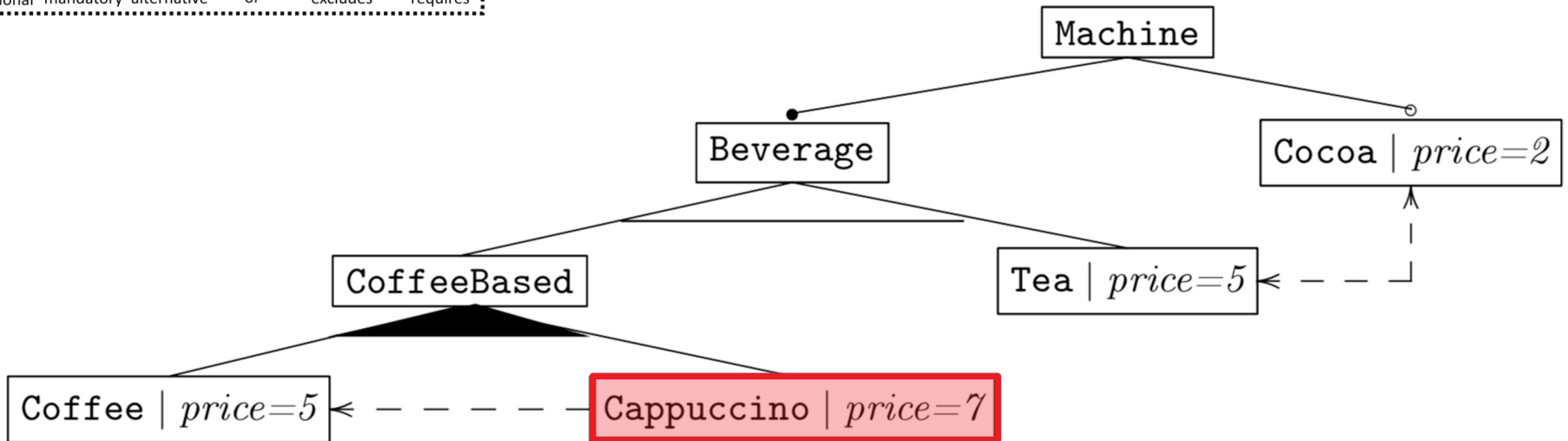
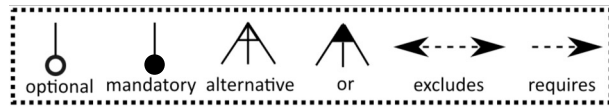
```
begin feature predicates
price= { Cappuccino = 7, Coffee = 5,
Cocoa = 2, Tea = 5 }
end feature predicates
```





# A simple vending machine product line

The feature model: Cross-tree constraints



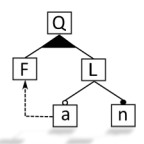
```
begin abstract features
Machine Beverage CoffeeBased
end abstract features
```

```
begin concrete features
Cocoa Tea Cappuccino Coffee
end concrete features
```

```
begin feature diagram
Machine -> {?Cocoa, Beverage}
Beverage -XOR-> {CoffeeBased, Tea}
CoffeeBased -OR-> {Cappuccino, Coffee}
end feature diagram
```

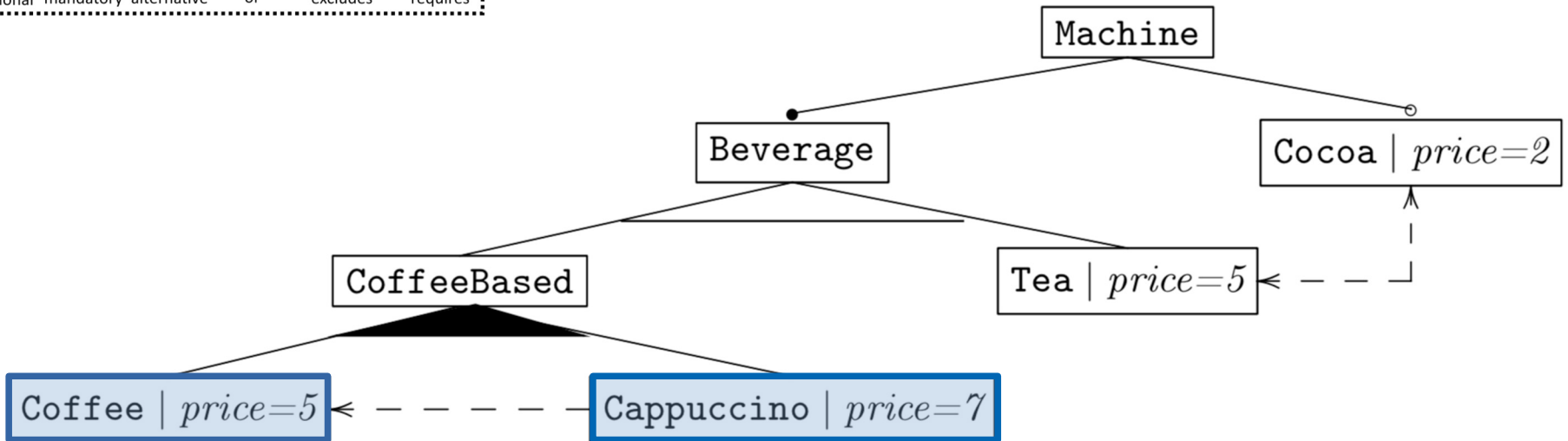
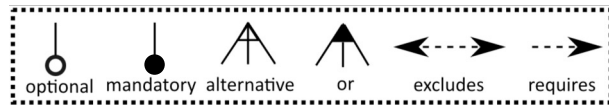
```
begin cross-tree constraints
Cappuccino requires Coffee
Tea excludes Cocoa
end cross-tree constraints
```

```
begin feature predicates
price= { Cappuccino = 7, Coffee = 5,
Cocoa = 2, Tea = 5 }
end feature predicates
```



# A simple vending machine product line

The feature model: Cross-tree constraints



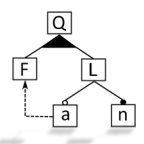
```
begin abstract features
Machine Beverage CoffeeBased
end abstract features
```

```
begin concrete features
Cocoa Tea Cappuccino Coffee
end concrete features
```

```
begin feature diagram
Machine -> {?Cocoa, Beverage}
Beverage -XOR-> {CoffeeBased, Tea}
CoffeeBased -OR-> {Cappuccino, Coffee}
end feature diagram
```

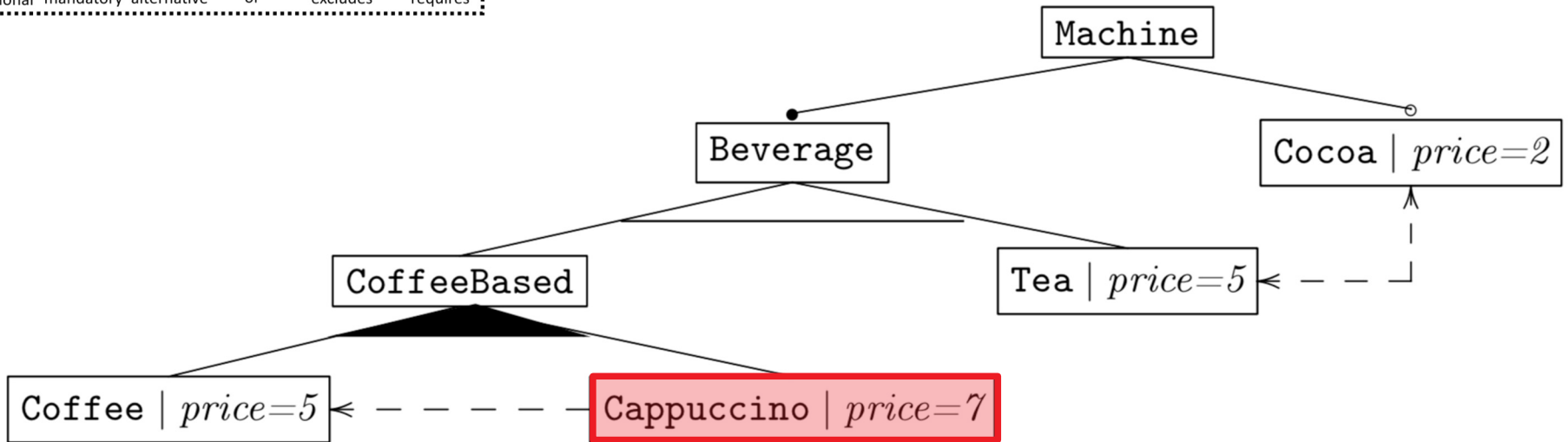
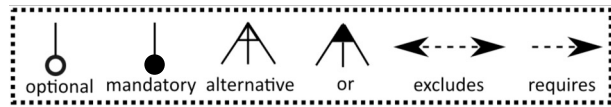
```
begin cross-tree constraints
Cappuccino requires Coffee
Tea excludes Cocoa
end cross-tree constraints
```

```
begin feature predicates
price= { Cappuccino = 7, Coffee = 5,
Cocoa = 2, Tea = 5 }
end feature predicates
```



# A simple vending machine product line

The feature model: Cross-tree constraints



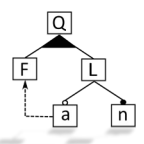
```
begin abstract features
Machine Beverage CoffeeBased
end abstract features
```

```
begin concrete features
Cocoa Tea Cappuccino Coffee
end concrete features
```

```
begin feature diagram
Machine -> {?Cocoa, Beverage}
Beverage -XOR-> {CoffeeBased, Tea}
CoffeeBased -OR-> {Cappuccino, Coffee}
end feature diagram
```

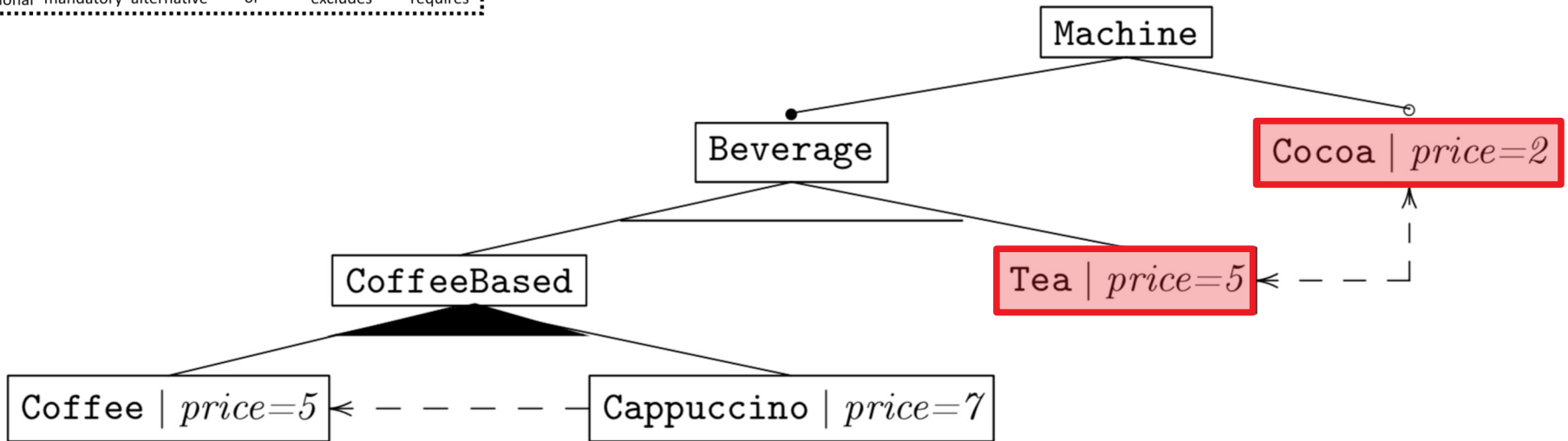
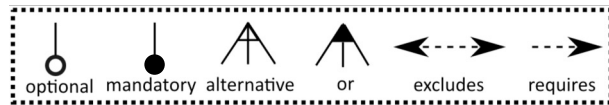
```
begin cross-tree constraints
Cappuccino requires Coffee
Tea excludes Cocoa
end cross-tree constraints
```

```
begin feature predicates
price= { Cappuccino = 7, Coffee = 5,
Cocoa = 2, Tea = 5 }
end feature predicates
```



# A simple vending machine product line

The feature model: Cross-tree constraints



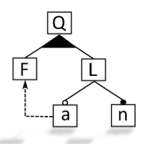
```
begin abstract features
Machine Beverage CoffeeBased
end abstract features
```

```
begin concrete features
Cocoa Tea Cappuccino Coffee
end concrete features
```

```
begin feature diagram
Machine -> {?Cocoa, Beverage}
Beverage -XOR-> {CoffeeBased,Tea}
CoffeeBased -OR->{Cappuccino,Coffee}
end feature diagram
```

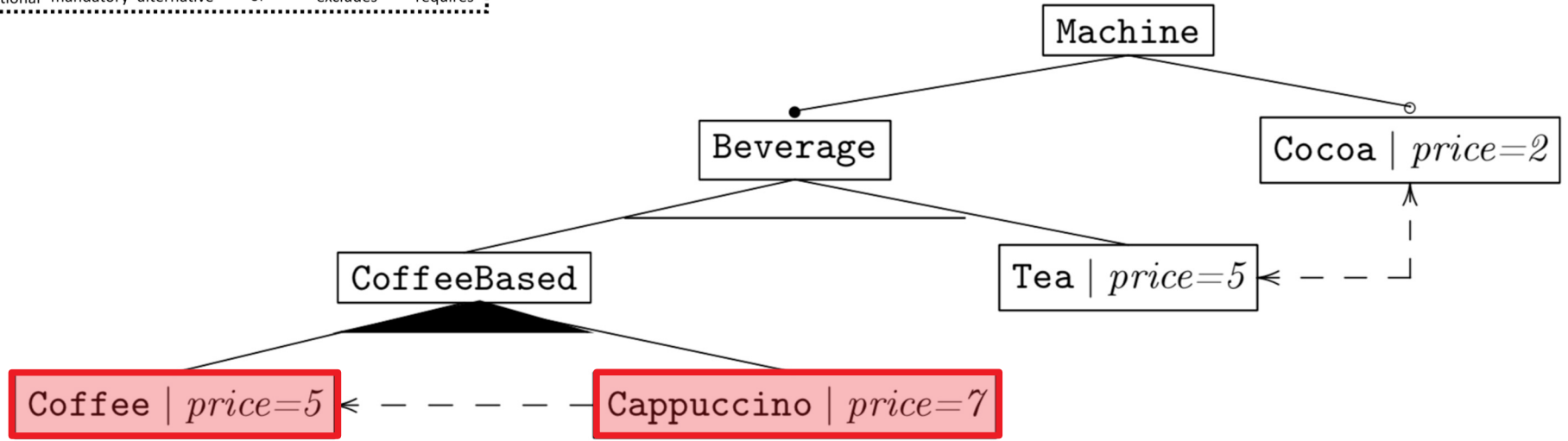
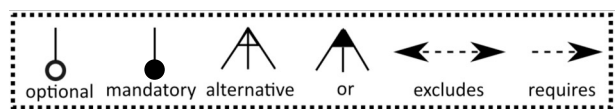
```
begin cross-tree constraints
Cappuccino requires Coffee
Tea excludes Cocoa
end cross-tree constraints
```

```
begin feature predicates
price= { Cappuccino = 7, Coffee = 5,
Cocoa = 2, Tea = 5 }
end feature predicates
```



# A simple vending machine product line

The feature model: Quantitative constraints



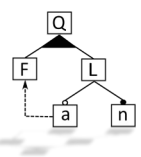
```
begin abstract features
Machine Beverage CoffeeBased
end abstract features
```

```
begin concrete features
Cocoa Tea Cappuccino Coffee
end concrete features
```

```
begin feature diagram
Machine -> {?Cocoa, Beverage}
Beverage -XOR-> {CoffeeBased, Tea}
CoffeeBased -OR-> {Cappuccino, Coffee}
end feature diagram
```

```
begin cross-tree constraints
Cappuccino requires Coffee
Tea excludes Cocoa
end cross-tree constraints
```

```
begin feature predicates
price= { Cappuccino = 7, Coffee = 5,
Cocoa = 2, Tea = 5 }
end feature predicates
begin quantitative constraints
{ price(Machine) <= 10 }
end quantitative constraints
```



## Feature Model

- Abstract and Concrete Features
- Cross-tree Constraints
- Quantitative Constraints

## Behaviour

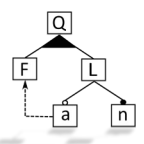
- Actions and Action Constraints
- Transitions
- Initial Configuration

## MultiVeStA Analysis

- Analysis when a condition holds
- Analysis at varying of time

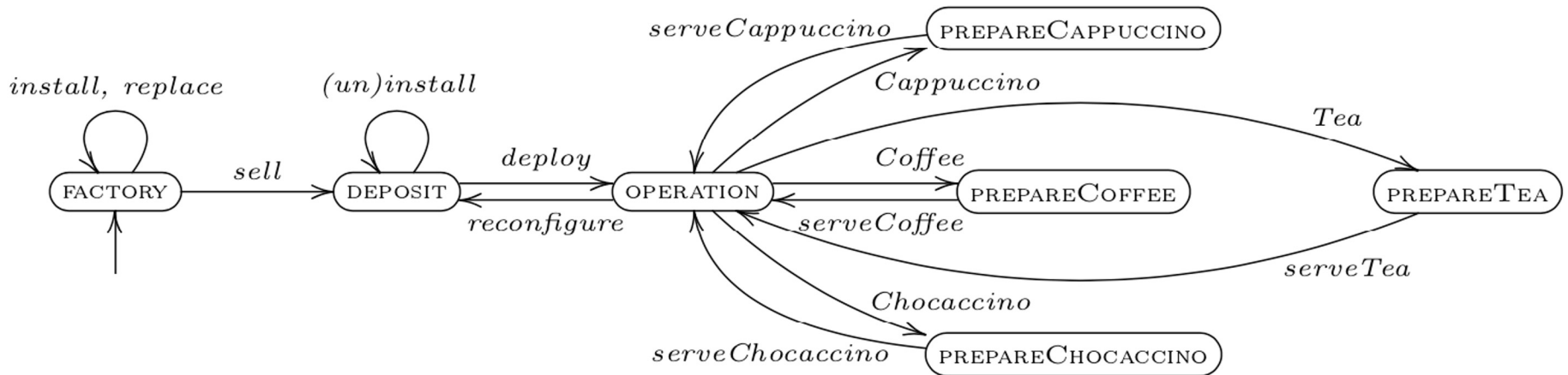
## An Application to a Simple Security Scenario

- Schneier's SafeLock Attack Tree



# A simple vending machine product line

Behaviour: actions and action constraints

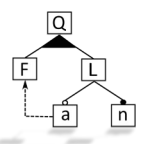


```

begin actions
  sell deploy reconfigure
  chocaccino
  serveCoffee serveCappuccino
  serveChocaccino serveTea
end actions
  
```

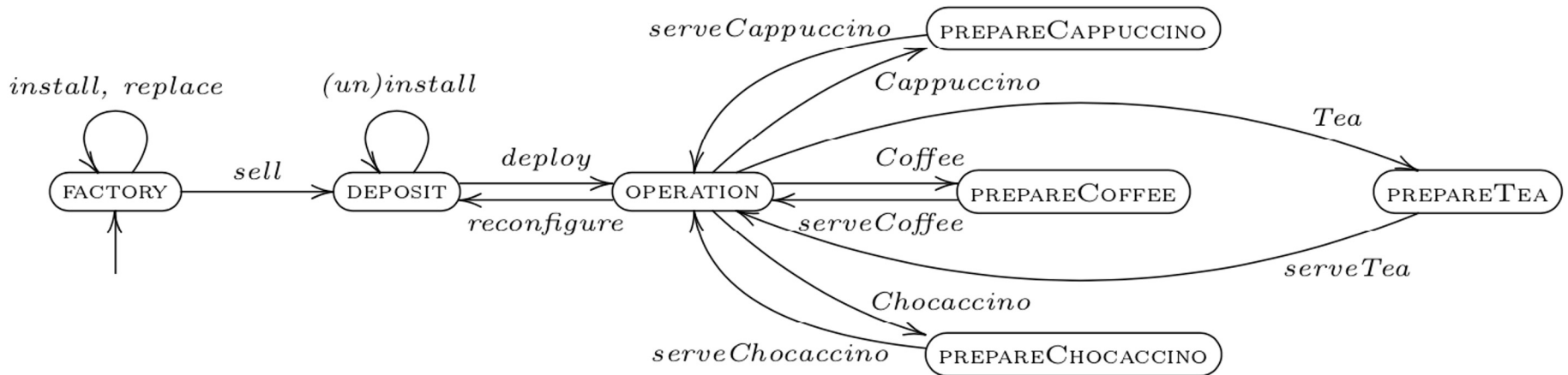
```

begin action constraints
  do(chocaccino) -> (has(Cappuccino)
    and has(Cocoa) )
end action constraints
  
```



# A simple vending machine product line

Behaviour: transitions



begin variables

sold = 0  
deploys = 0

end variables

begin actions

sell deploy reconfigure  
chocaccino  
serveCoffee serveCappuccino  
serveChocaccino serveTea

end actions

begin action constraints

do(chocaccino) -> (has(Cappuccino)  
and has(Cocoa) )

end action constraints

begin processes diagram

begin process dynamics

states = factory , deposit , operating , prepareCoffee ,  
prepareCappuccino, prepareTea , prepareChocaccino

//Operating

//Coffee

operating -(Coffee,3)-> prepareCoffee,  
prepareCoffee -(serveCoffee,1) -> operating,

//Cappuccino

operating -(Cappuccino,3)-> prepareCappuccino,  
prepareCappuccino -(serveCappuccino,1) -> operating,

//Chocaccino

operating -(chocaccino,2)-> prepareChocaccino,  
prepareChocaccino -(serveChocaccino,1) -> operating,

//Tea

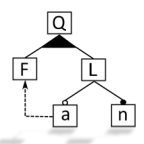
operating -(Tea,3)-> prepareTea,  
prepareCappuccino -(serveTea,1) -> operating,

operating -(reconfigure,1) -> deposit

end process

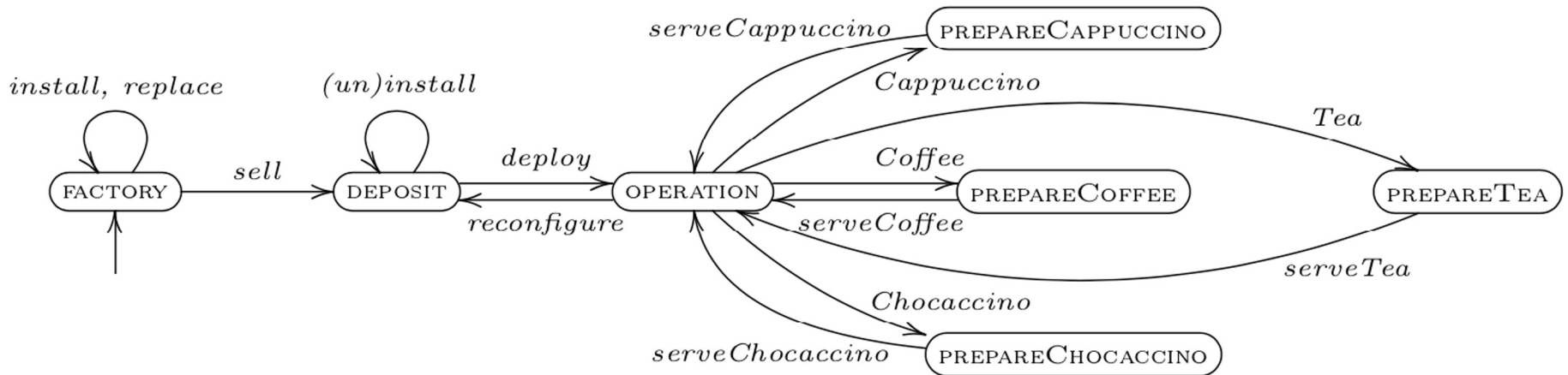
end processes diagram





# A simple vending machine product line

## Behaviour: initial configuration



```

begin variables
sold = 0
deloys = 0
end variables
begin actions
sell deploy reconfigure
chocaccino
serveCoffee serveCappuccino
serveChocaccino serveTea
end actions
begin action constraints
do(chocaccino) -> (has(Cappuccino)
and has(Cocoa) )
end action constraints
begin init
installedFeatures = { Coffee }
initialProcesses = dynamics
end init

```

```

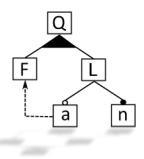
begin processes diagram
begin process dynamics
states = factory , deposit , operating , prepareCoffee ,
prepareCappuccino , prepareTea , prepareChocaccino
transitions =
//Factory
factory -(replace(Coffee,Tea),2.0)->factory,
factory -(install(Cocoa),1.0)->factory,
factory -(install(Cappuccino),1.0)->factory,
factory -(sell,1,{sold=1})-> deposit,
//Deposit
deposit -(install(Cappuccino),2.0)->deposit,
deposit -(uninstall(Cappuccino),2.0)->deposit,
deposit -(install(Cocoa),2.0)->deposit,
deposit -(uninstall(Cocoa),2.0)->deposit,
deposit -(deploy,2,{deloys=deloys+1})-> operating

```

```

//Operating
//Coffee
operating -(Coffee,3)-> prepareCoffee,
prepareCoffee -(serveCoffee,1) -> operating,
//Cappuccino
operating -(Cappuccino,3)-> prepareCappuccino,
prepareCappuccino -(serveCappuccino,1) -> operating,
//Chocaccino
operating -(chocaccino,2)-> prepareChocaccino,
prepareChocaccino -(serveChocaccino,1) -> operating,
//Tea
operating -(Tea,3)-> prepareTea,
prepareCappuccino -(serveTea,1) -> operating,
operating -(reconfigure,1) -> deposit
end process
end processes diagram

```



## Feature Model

- Abstract and Concrete Features
- Cross-tree Constraints
- Quantitative Constraints

## Behaviour

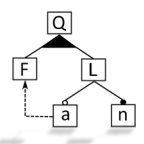
- Actions and Action Constraints
- Transitions
- Initial Configuration

## **MultiVeStA Analysis**

- Analysis when a condition holds
- Analysis at varying of time

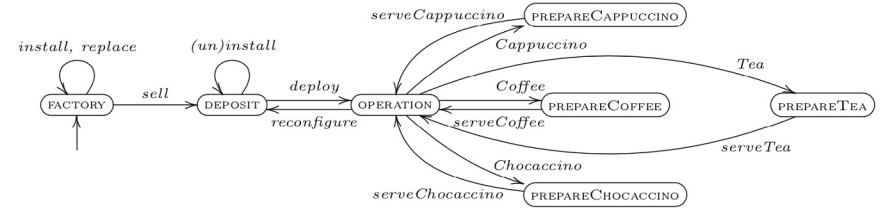
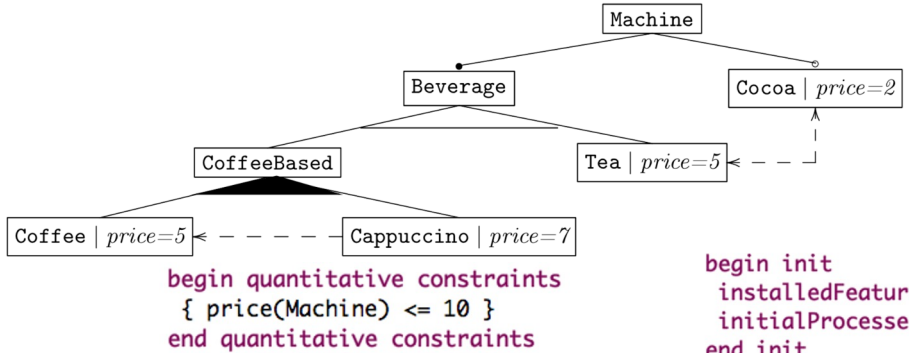
## An Application to a Simple Security Scenario

- Schneier's SafeLock Attack Tree



# A simple vending machine product line

## MultiveSTA Analysis: analysis of sold machines



begin init  
 installedFeatures = { Coffee }  
 initialProcesses = dynamics  
 end init

begin variables  
 sold = 0  
 deploys = 0  
 end variables

begin action constraints  
 do(chocaccino) -> (has(Cappuccino)  
 and has(Cocoa) )  
 end action constraints

begin analysis

```

query = eval when {sold == 1.0 } :
{ price(Machine) [delta=0.5],
  Coffee , Tea , Cappuccino , Cocoa
}
  
```

```

default delta=0.05
alpha = 0.05
parallelism = 1
  
```

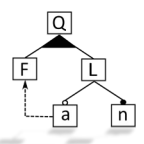
end analysis

begin processes diagram  
 begin process dynamics

states = factory , deposit , operating , prepareCoffee ,  
 prepareCappuccino, prepareTea , prepareChocaccino

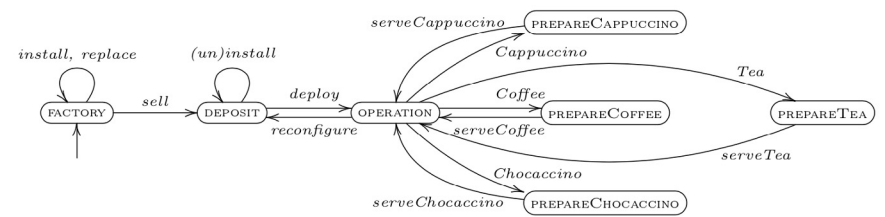
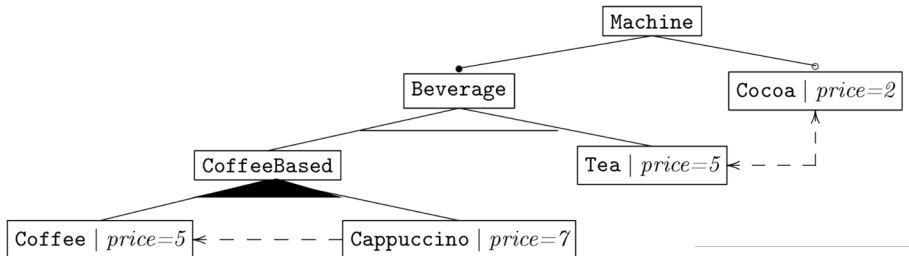
```

transitions =
//Factory
factory -(replace(Coffee,Tea),20)->factory,
factory -(install(Cocoa),10)->factory,
factory -(install(Cappuccino),10)->factory,
factory -(sell,1,{sold=1})-> deposit,
//Deposit
deposit -(install(Cappuccino),2.0)->deposit,
deposit -(uninstall(Cappuccino),2.0)->deposit,
deposit -(install(Cocoa),2.0)->deposit,
deposit -(uninstall(Cocoa),2.0)->deposit,
deposit -(deploy,2,{deploys=deploys+1})-> operating ,
  
```



# A simple vending machine product line

## MultiveSTA Analysis: analysis of sold machines



```
begin quantitative constraints
{ price(Machine) <= 15 }
end quantitative constraints
```

```
begin init
  installedFeatures = { Coffee }
  initialProcesses = dynamics
end init
```

```
begin variables
  sold = 0
  deploys = 0
end variables
```

```
begin action constraints
  do(chocaccino) -> (has(Cappuccino)
                    and has(Cocoa) )
end action constraints
```

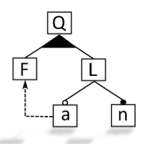
begin analysis

```
query = eval when {sold == 1.0 } :
{ price(Machine) [delta=0.5],
  Coffee , Tea , Cappuccino , Cocoa
}
```

```
default delta=0.05
alpha = 0.05
parallelism = 1
```

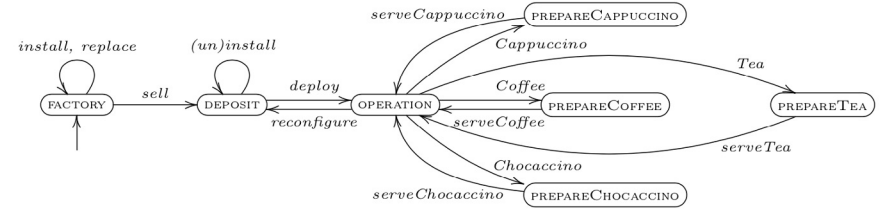
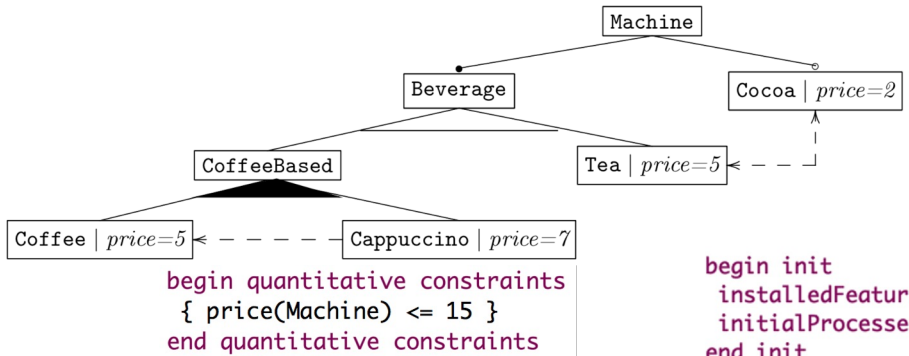
end analysis

	Price	Coffee	Tea	Cappucino	Cocoa
	5.68	0.36	0.64	0.00	0.34



# A simple vending machine product line

## MultiVeStA Analysis: analysis at varying of time



```

begin init
  installedFeatures = { Coffee }
  initialProcesses = dynamics
end init
  
```

```

begin variables
  sold = 0
  deploys = 0
end variables
  
```

```

begin action constraints
  do(chocaccino) -> (has(Cappuccino)
                    and has(Cocoa) )
end action constraints
  
```

begin analysis

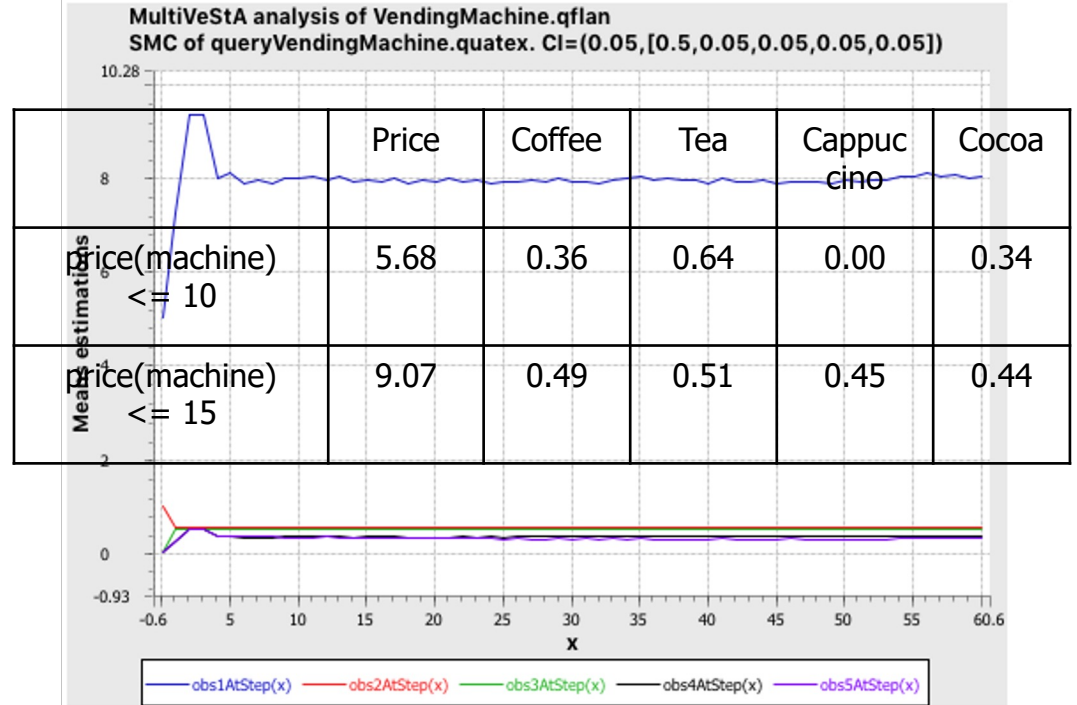
```

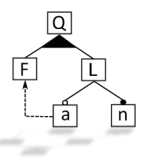
query = eval when_{sold == 1.0} :
{ price(Machine) [delta=0.5],
  Coffee , Tea , Cappuccino , Cocoa
}
  
```

```

default delta=0.05
alpha = 0.05
parallelism = 1
  
```

end analysis





## Feature Model

- .Abstract and Concrete Features
- .Cross-tree Constraints
- .Quantitative Constraints

## Behaviour

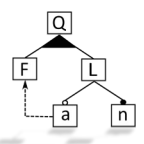
- .Actions and Action Constraints
- .Transitions
- .Initial Configuration

## MultiVeStA Analysis

- Analysis when a condition holds
- Analysis at varying of time

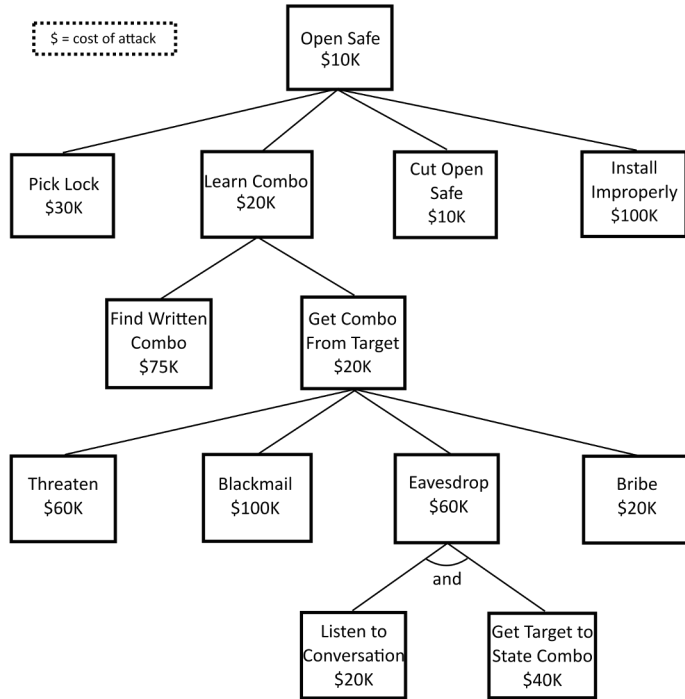
## **An Application to a Simple Security Scenario**

- Schneier's SafeLock Attack Tree



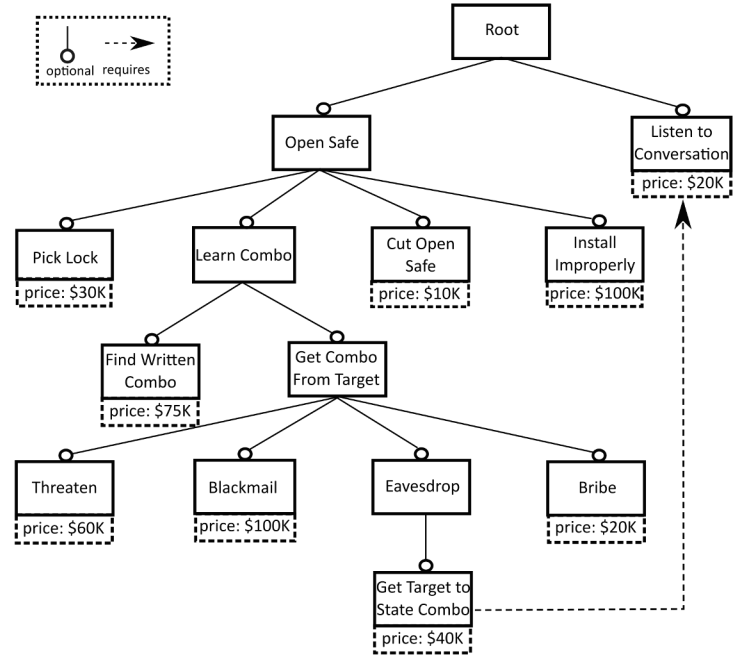
# Schneier's SafeLock Attack Tree

An application of QFLan to security



Schneier's simple attack tree

[www.schneier.com/academic/archives/1999/12/attack\\_trees.html](http://www.schneier.com/academic/archives/1999/12/attack_trees.html)



A feature model version of the attack tree  
[TSE'18]

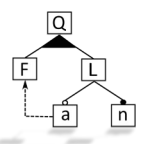
*install(PickLock) ... install(Bribe)*



begin quantitative constraints

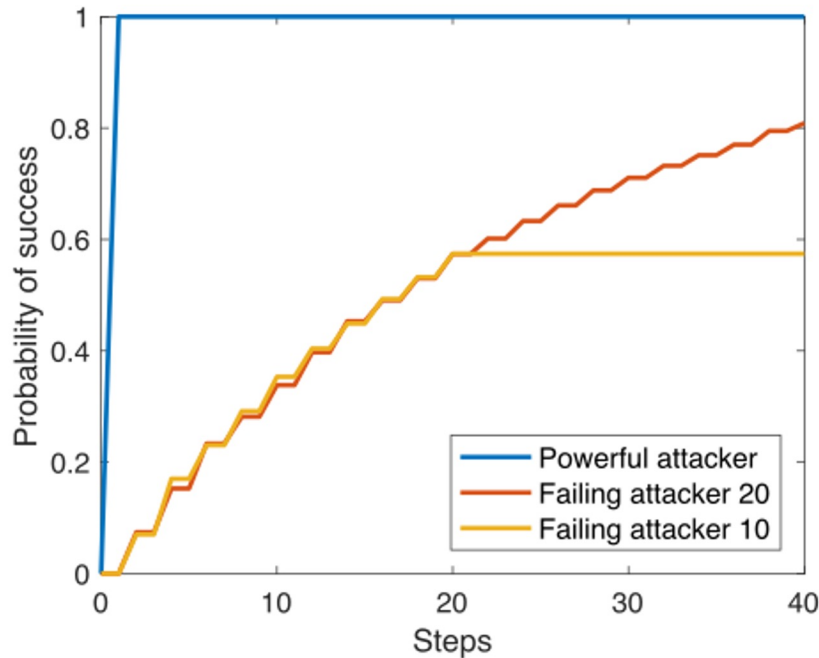
```
//Restrict to attacks that cost less than 100
{ cost(Root) <= 100 }
```

end quantitative constraints

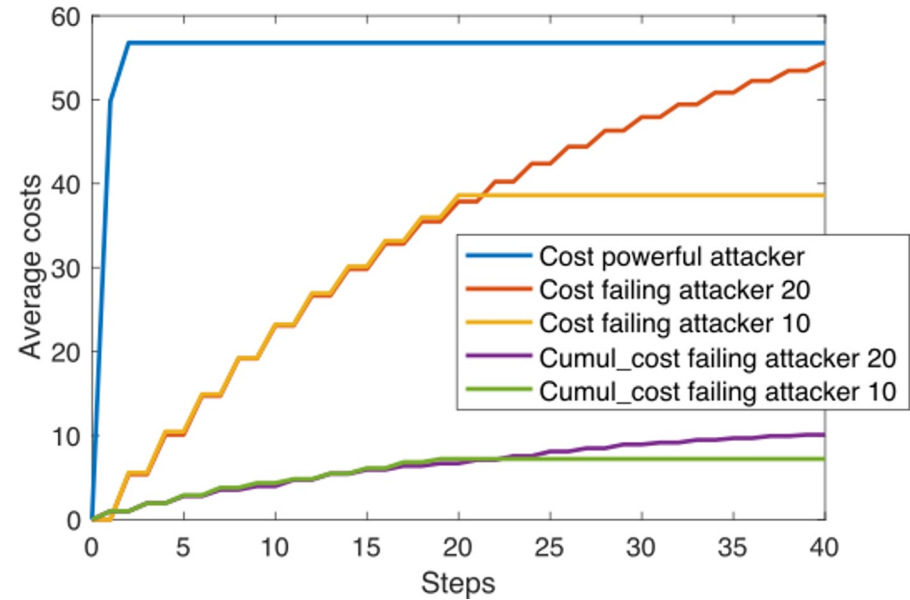


# Schneier's SafeLock Attack Tree

An application of QFLan to security

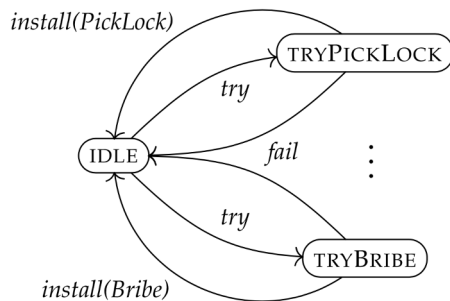


Probabilities of successful attacks



Costs of successful attacks

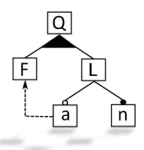
*install(PickLock) ... install(Bribe)*



**begin quantitative constraints**

```
//Restrict to attacks that cost less than 100
{ cost(Root) <= 100 }
//Attacks can fail. Attacks attempts cost.
//Restrict to attackers with a maximum budget.
{ accumulated_cost <= 10 }
//{ accumulated_cost <= 20 }
end quantitative constraints
```





Extend semantics with notion of time

For the analysis of time-related properties

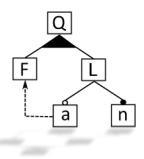
**Continue investigating applicability to security domain**

**Adapt QFLan to attack trees domain**

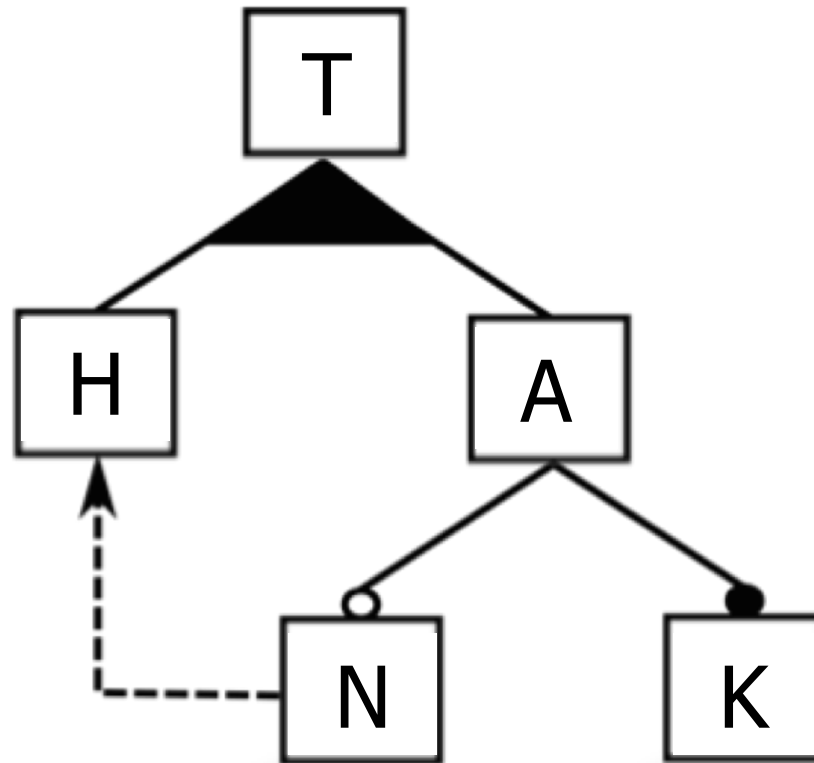
Synthesis of constraints

We had to relax the constraint "price(Machine)  $\leq 10$ "

Can we synthesize the 'right' constraints automatically?

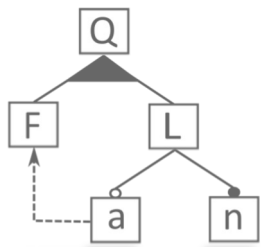


QUESTIONS?

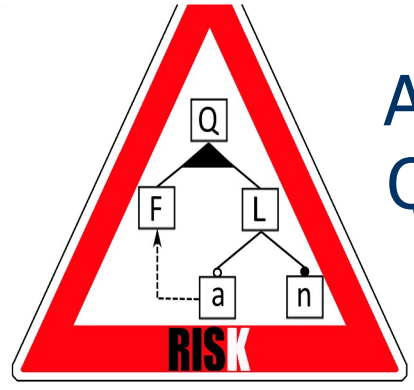


**YOU!**

<https://github.com/qflanTeam/QFLan/>



QFLan: A tool for the  
Quantitative Analysis of  
Highly Reconfigurable Systems



## A Software Engineering Approach to Quantitative Security Risk Modeling and Analysis using QFLan

Andrea Vandin

Sant'Anna School of Advanced Studies Pisa, Italy  
DTU Technical University of Denmark

Maurice H. ter Beek  
ISTI CNR Pisa, Italy

Axel Legay  
UCLouvain, Belgium

Alberto Lluch Lafuente  
DTU, Denmark

Classes 21t-22t, Software Validation and Verification, Unipi, 04-05/12/2023  
Class 21t 04/12/2023

## From QFLan to RisQFLan

- QFLan's Limitations for Risk Modeling and Analysis
- A Bank robbery scenario in RisQFLan
- How did we go from QFLan to RisQFLan?

## Conclusions

From QFLan to RisQFLan

- **QFLan's Limitations for Risk Modeling and Analysis**
- A Bank robbery scenario in RisQFLan
- How did we go from QFLan to RisQFLan?

Conclusions

Not entirely direct encoding of the scenario

- The extra root node, the extra states to model failures, etc

We need different types of nodes

- Attack, defense, countermeasures

We need richer constructs

- QFLan has: or, requires, excludes

- Missing *common* constructs: and

Attack attempts might fail

- The 'install' of an attack node requires

There is no 'absolute security'

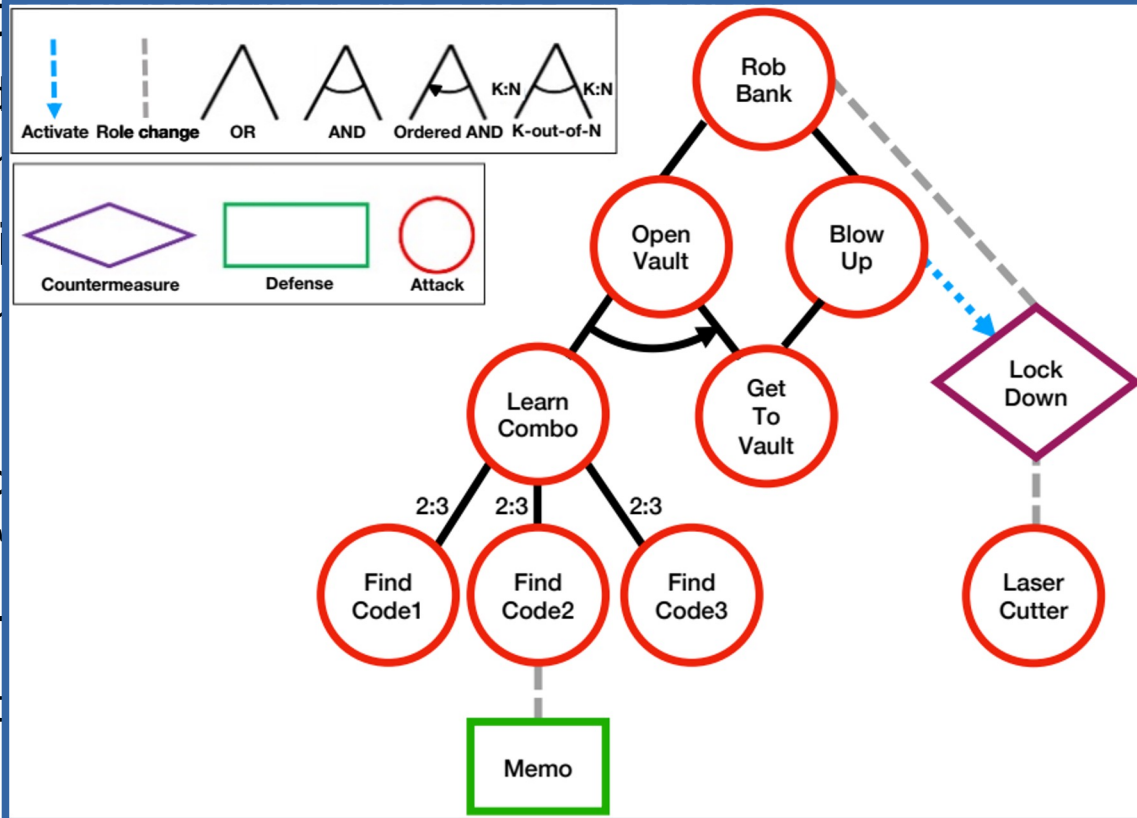
- *Qualitative* constraints like 'exclusive'

- Often, failure probabilities are very low

Exact analysis might be required

- Complement MultiVeStA Statistics

We want to model scenarios like this



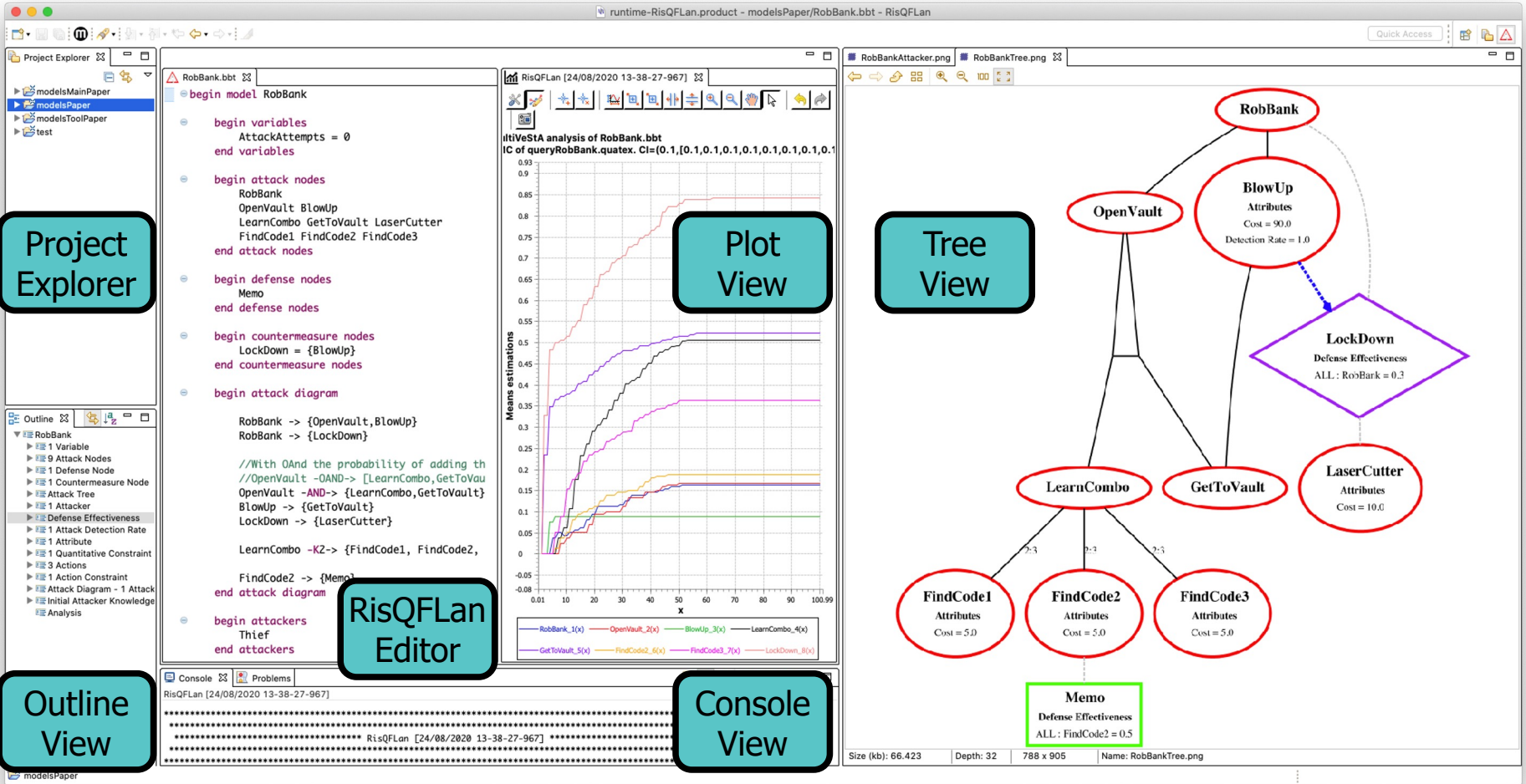
## From QFLan to RisQFLan

- QFLan's Limitations for Risk Modeling and Analysis
- **A Bank robbery scenario in RisQFLan**
- How did we go from QFLan to RisQFLan?

## Conclusions

# A Bank Robbery Scenario in RisQFLan

## A screenshot of RisQFLan



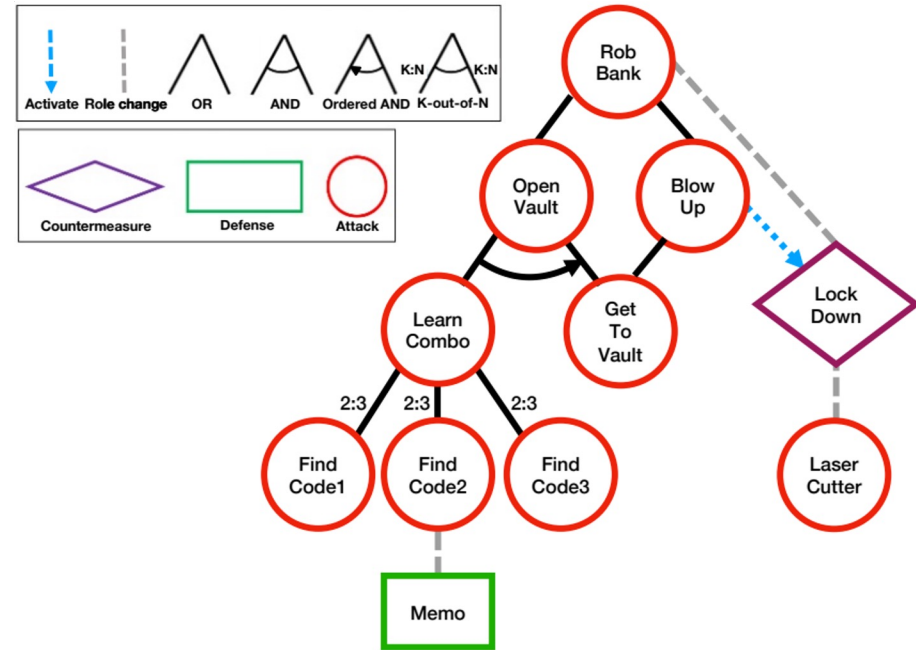
The screenshot displays the RisQFLan software interface with several key components highlighted by callouts:

- Project Explorer:** Shows the project structure including modelsMainPaper, modelsPaper, modelsToolPaper, and test.
- RisQFLan Editor:** Contains the model definition for RobBank, including variables, attack nodes, defense nodes, countermeasure nodes, and an attack diagram. The attack diagram shows a sequence of actions: RobBank attacks OpenVault and BlowUp, which leads to LearnCombo and GetToVault, and finally to LockDown. Countermeasures include Memo and LaserCutter.
- Plot View:** Displays a graph titled "ittiveSTA analysis of RobBank.bbt" showing mean estimations over time (x-axis) for various nodes like RobBank\_1(x), OpenVault\_2(x), BlowUp\_3(x), LearnCombo\_4(x), GetToVault\_5(x), FindCode2\_6(x), FindCode3\_7(x), and LockDown\_8(x).
- Tree View:** A decision tree diagram starting with RobBank, branching into OpenVault and BlowUp. OpenVault leads to LearnCombo, which further branches into FindCode1, FindCode2, and FindCode3. BlowUp leads to GetToVault and LaserCutter. A LockDown node is also shown, influenced by BlowUp and FindCode2.
- Outline View:** Provides a hierarchical overview of the model's structure, including variables, attack nodes, defense nodes, countermeasure nodes, and the attack diagram.
- Console View:** Shows the execution log and output of the simulation.

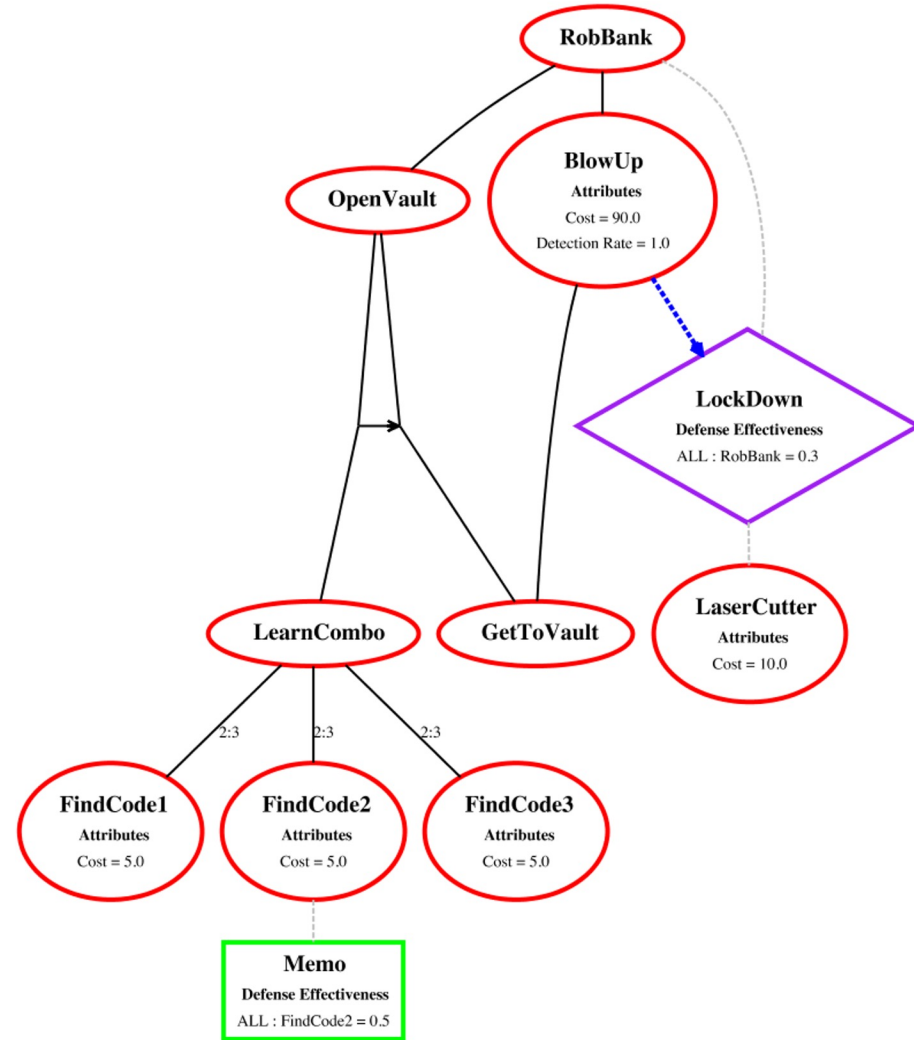


# A Bank Robbery Scenario in RisQFLan

## Attack-defense tree

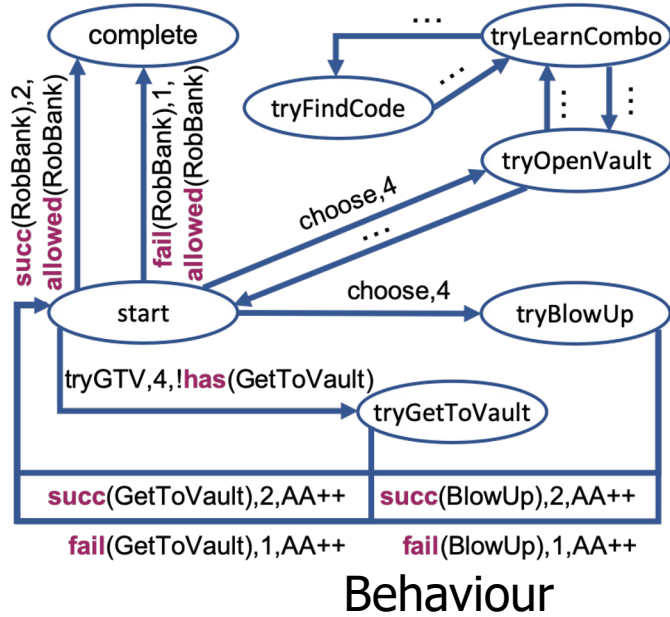


Desired scenario

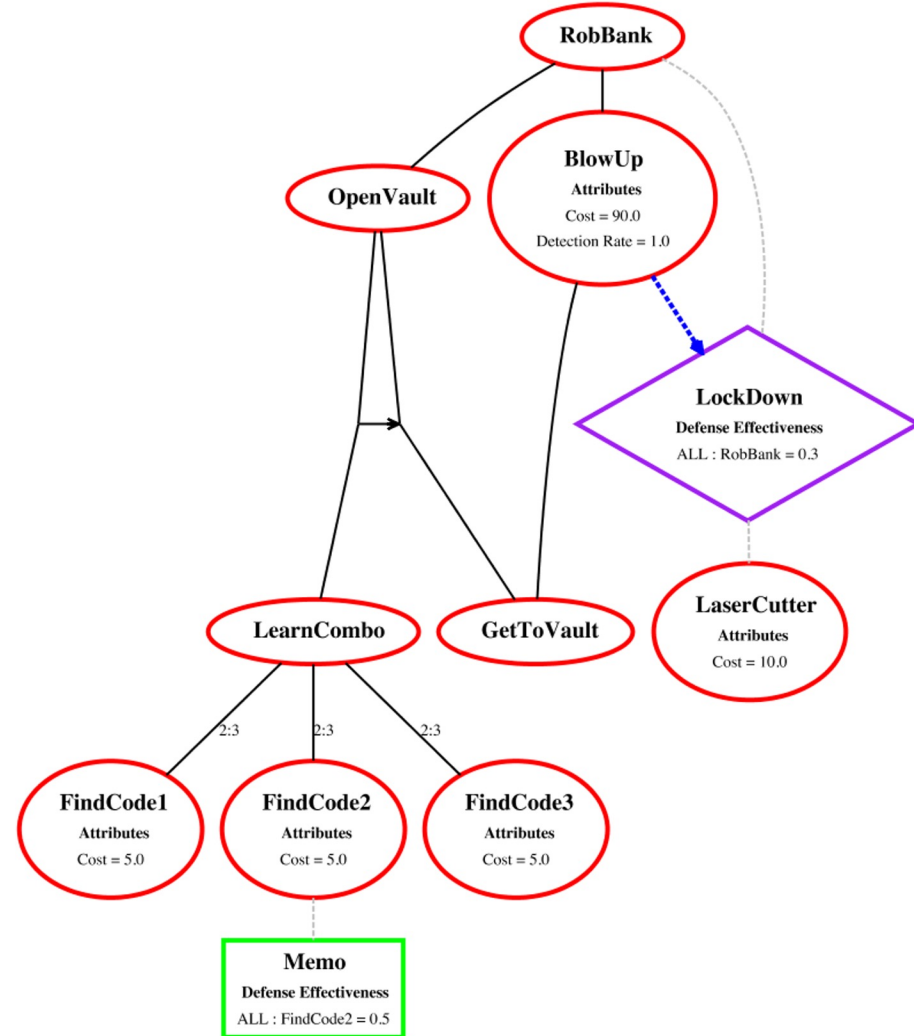


Modeled Scenario

# A Bank Robbery Scenario in RisQFLan Behaviour



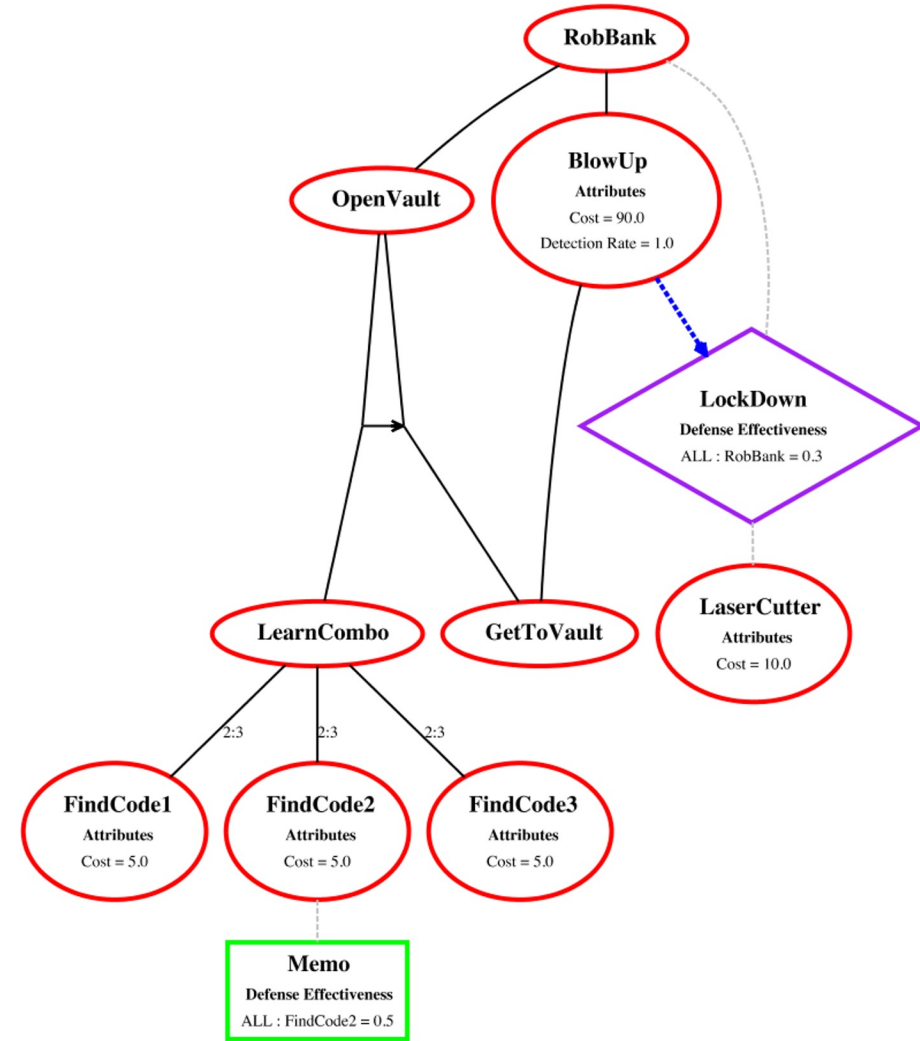
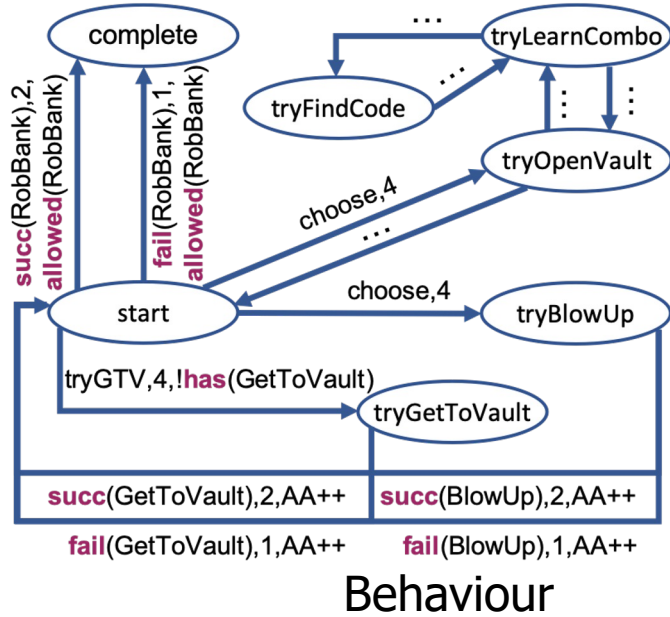
Behaviour



Attack-defense tree

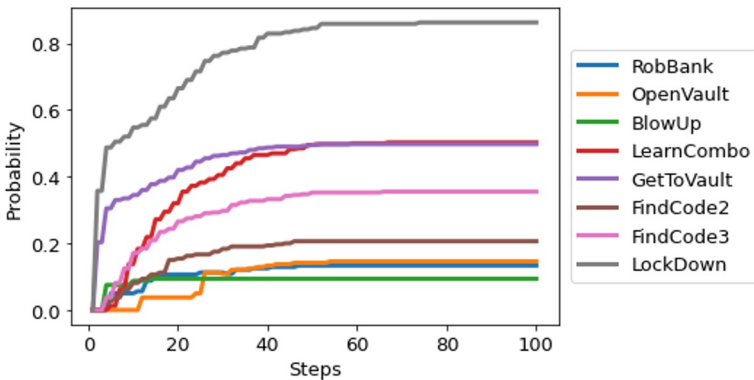
# A Bank Robbery Scenario in RisQFLan

## Analysis: SMC with MultiVeStA



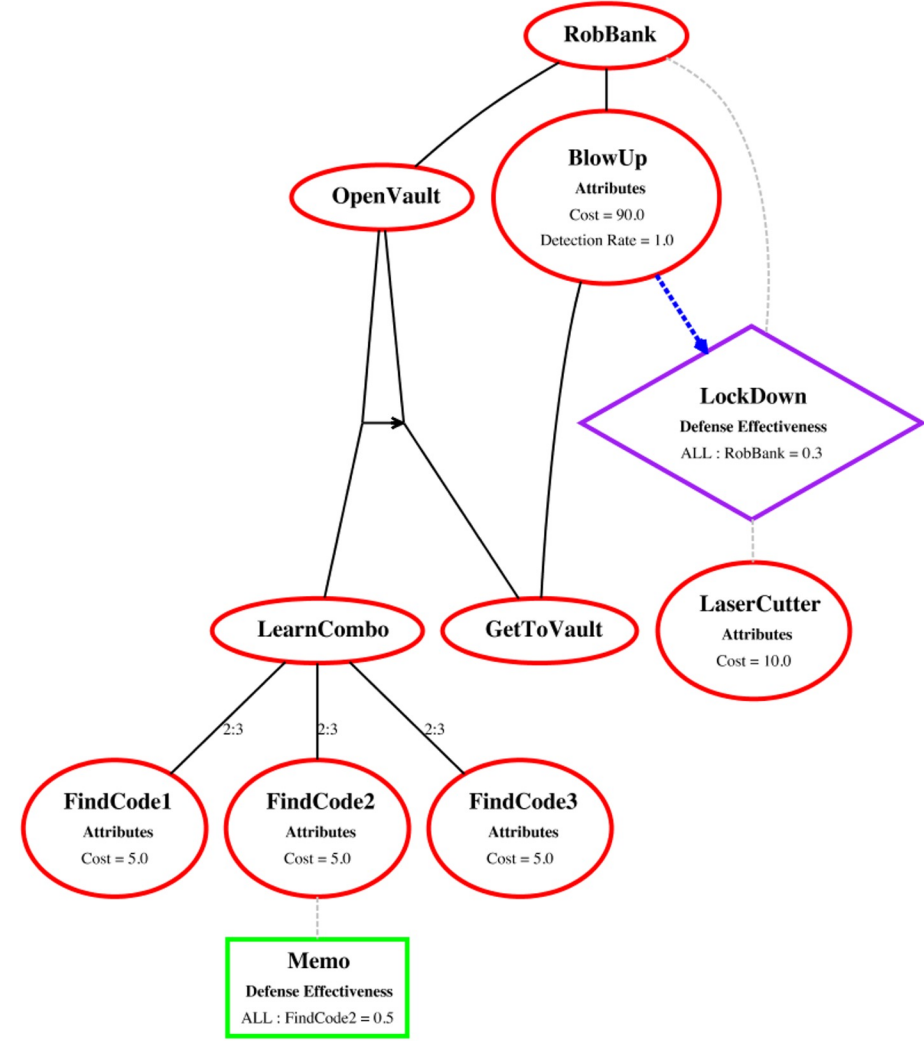
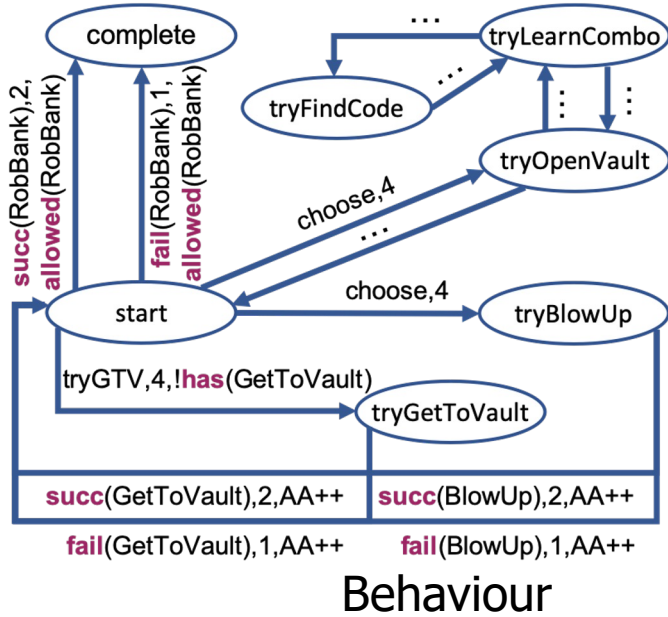
```
begin analysis
query = eval from 1 to 100 by 1 :
{ RobBank, OpenVault, BlowUp,
  LearnCombo, GetToVault,
  FindCode2, FindCode3, LockDown }
default delta = 0.1 alpha = 0.1
parallelism = 1
end analysis
```

### Statistical SMC Analysis



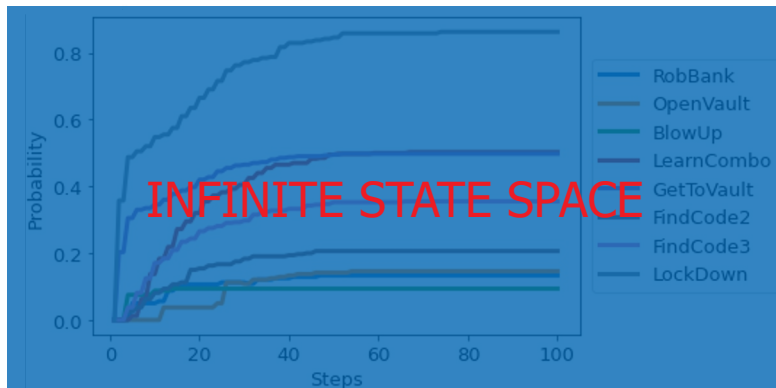
# A Bank Robbery Scenario in RisQFLan

## Analysis: PMC with PRISM/STORM



```
begin exportDTMC
  file = "RobBank.prism"
  label with "Succeeded"
  when has(RobBank)
end exportDTMC
```

Exact  
PMC Analysis



## From QFLan to RisQFLan

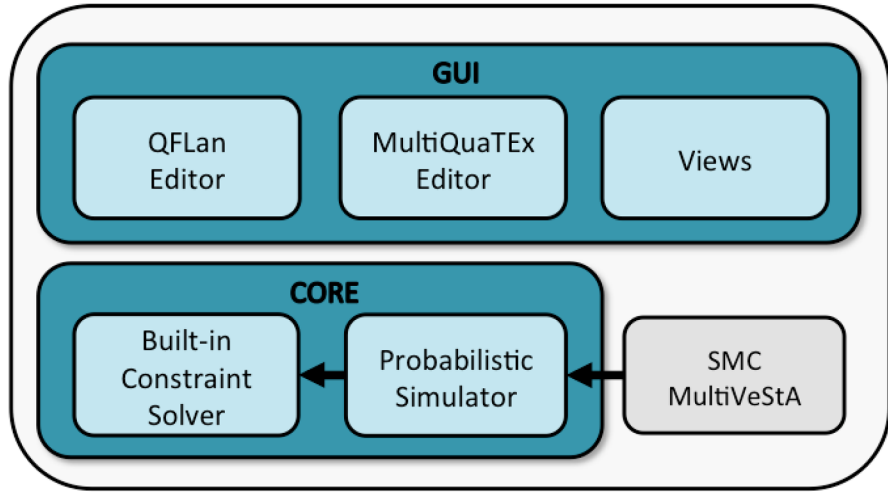
- QFLan's Limitations for Risk Modeling and Analysis
- A Bank robbery scenario in RisQFLan
- **How did we go from QFLan to RisQFLan?**

Conclusions

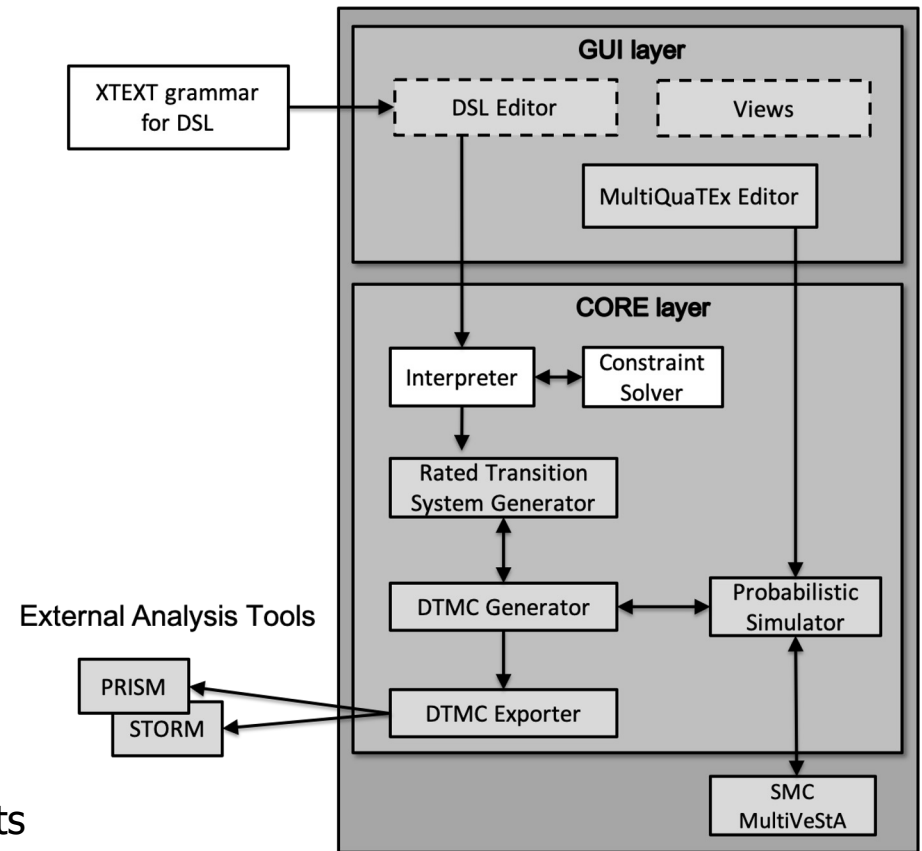
# From QFLan to RisQFLan

## Generalizing the QFLan approach

### QFLan Architecture [FM'18][TSE'18]



### Generalized QFLan Architecture [Draft'20]



Existing domain-independent components

Automatically generated domain-independent components

Domain-specific components necessary to instantiate the architecture in a new domain

## From QFLan to RisQFLan

- QFLan's Limitations for Risk Modeling and Analysis
- A Bank robbery scenario in RisQFLan
- How did we go from QFLan to RisQFLan?

## **Conclusions**

## **RisQFLan: A Software Engineering Approach to Quantitative Security Risk Modeling and Analysis**

- Obtained via a DSL-independent generalization of QFLan + its instantiation to security domain
- Both QFLan and RisQFLan are open-source projects

### **Main improvements**

- Modeling: Richer constructs specific to the security domain
- Analysis: New support for exact PMC engines (**PRISM**, STORM) complementing existing SMC engine (MultiVeStA)

### **Related work**

- Due to the generality and versatility of our framework, we succeeded in incorporating many features from proposals in the literature
- E.g.: o-and, noticeability, countermeasures (see validation in [Draft20])
- The explicit probabilistic attacker behaviour is somehow new, as
- Specific dynamic threat profiles is a related feature. But it is often unsupported
  - Supported only recently by a few approaches in a limited way
- RisQFLan allows for nodes with multiple parents
  - This is convenient: allows to keep models small. But it is often unsupported



Attributes of leaf nodes are propagated up the tree via sum.

- Other approaches, e.g. SecurITree, allow for attribute-specific propagation functions (e.g., min, max, product)

Allow for non-deterministic (unspecified) aspects in RisQFLan

- Use external tools (Uppaal Stratego?) to synthesize the attacker with highest success probability/the defense with best impact

Even though the design of RisQFLan is inspired by the most common features from the literature, we want to:

- Better understand relation of RisQFLan with the huge related work

Validate RisQFLan scalability and expressiveness considering realistic scenarios

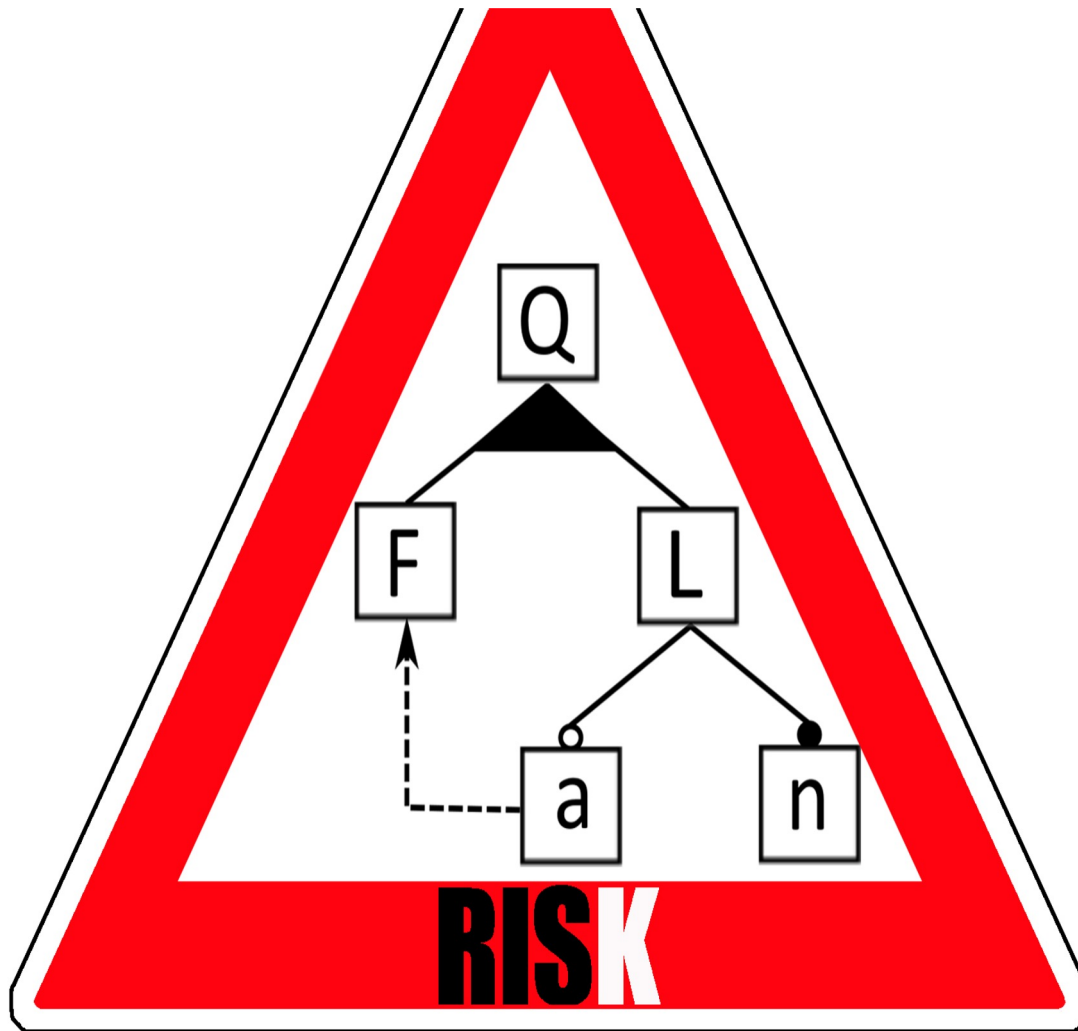
- E.g. the Attack Tree Benchmarks [www7.in.tum.de/~kraemerj/upload/index.php](http://www7.in.tum.de/~kraemerj/upload/index.php)

The great expressive power coming from the quantitative constraints, etc, might make it difficult to understand what a model does

SMC and PMC give only limited information on what the model does

- We get black-box numbers
- Are these numbers due to the nature of the studied system?
- Are these numbers due to bugs?

Can we exploit novel techniques to **explain SMC?**



---

# STATISTICAL MODEL CHECKING MEETS PROCESS MINING

WHITE-BOX VALIDATION OF SIMULATION MODELS

---

Andrea Vandin



**Sant'Anna**

School of Advanced Studies – Pisa

Institute of Economics



Department  
of Excellence  
2018 - 2022

*2023-2027!*

**EMbeDS**

Economics and Management  
in the era of Data Science



Danmarks  
Tekniske  
Universitet

Classes 21t-22t, Software Validation and Verification, Unipi, 04-05/12/2023

Class 21t 04/12/2023

# WHITE-BOX VALIDATION OF PRODUCT LINES AND THREAT MODELS BY STATISTICAL MODEL CHECKING AND PROCESS MINING

---

Roberto Casaluce, Andrea Burattin, Francesca Chiaromonte,  
Alberto Lluch Lafuente, Andrea Vandin

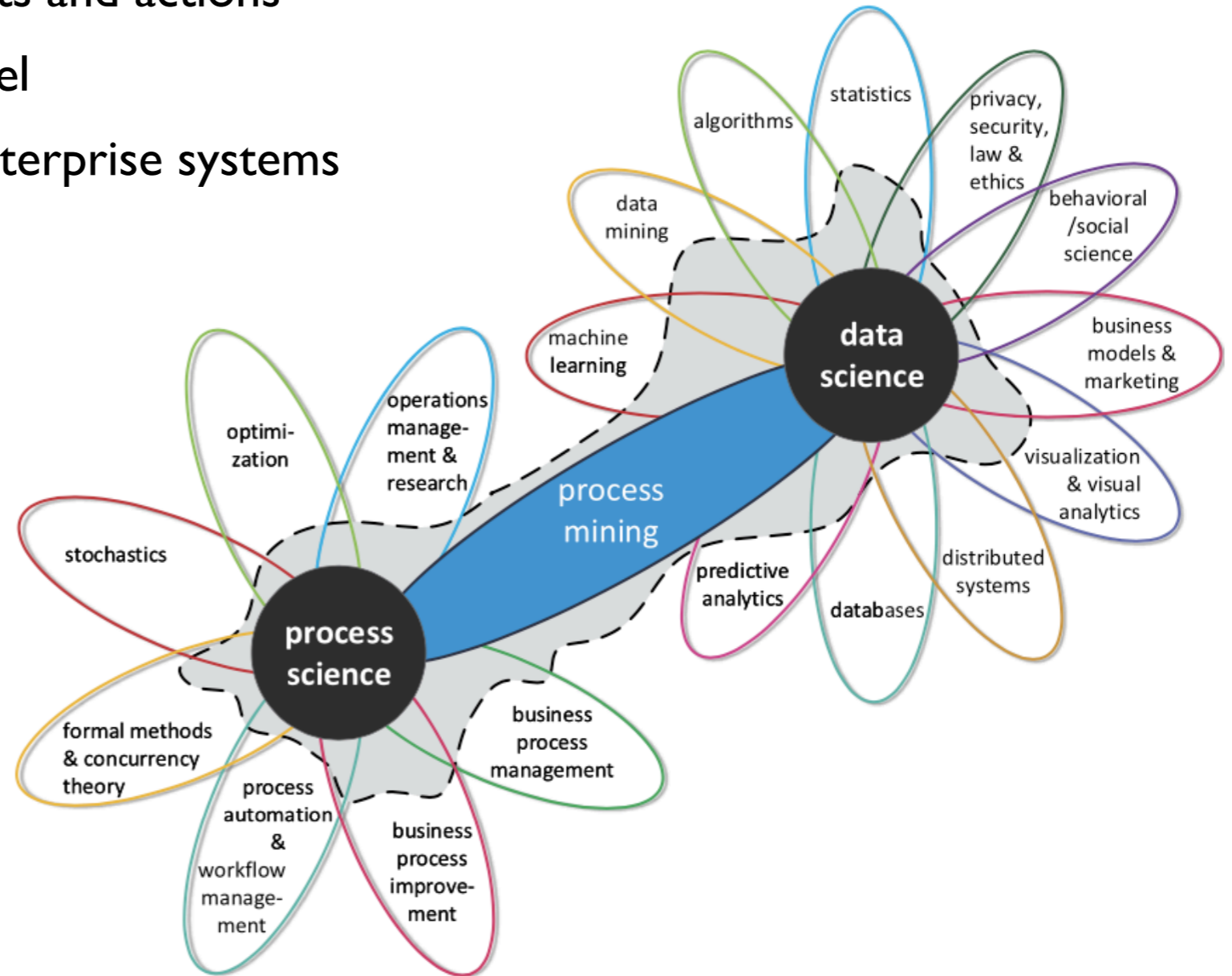
Recently published at DEC2H  
[zenodo.org/record/6623377](https://zenodo.org/record/6623377)

Journal extension at JSS: 2nd round of review  
[zenodo.org/record/6623377](https://zenodo.org/record/6623377)

'A SIMPLIFIED OVERVIEW'

# WHAT IS PROCESS MINING?

- A family of techniques linking data science and process management to support the analysis of processes
- Aims at turning event logs into insights and actions
- Uses data to discover a process model
  - It observes events recorded by enterprise systems



Van Der Aalst, W., et al. (2011, August).

Process mining manifesto. In *Conference on Business Process Management*

# WHAT IS PROCESS MINING?

---



Picture by Koen Olsthoorn

The reference process...

The traces, or process logs  
▶ The *actual* process...

# WHAT IS PROCESS MINING?



Picture by Koen Olsthoorn

With Process Mining we can discover that

- The actual process is different from the expected one

**Idea**

**Can PM explain the SMC results?**

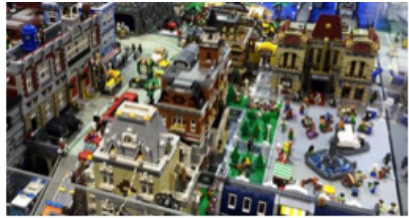
The reference process...

The traces, or process logs

- The *actual* process...



# OUR METHODOLOGY 4 WHITE-BOX VALIDATION

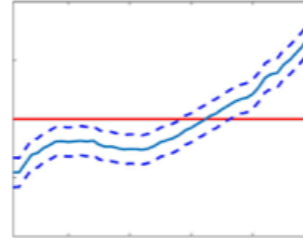


[newstalkzb.co.nz/news/education/modern-lego-sets-more-complex-less-inspiring/](http://newstalkzb.co.nz/news/education/modern-lego-sets-more-complex-less-inspiring/)

Query  $\alpha, \delta$  Statistical confidence

**Statistical Model Checking**



- ▶ Automatic
- ▶ Time-saving and Reproducible
- ▶ Promotes use of *standard* analysis
- ▶ Reference implementation
- ▶ Reliable and Efficient



# OUR METHODOLOGY 4 WHITE-BOX VALIDATION

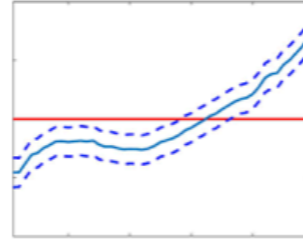
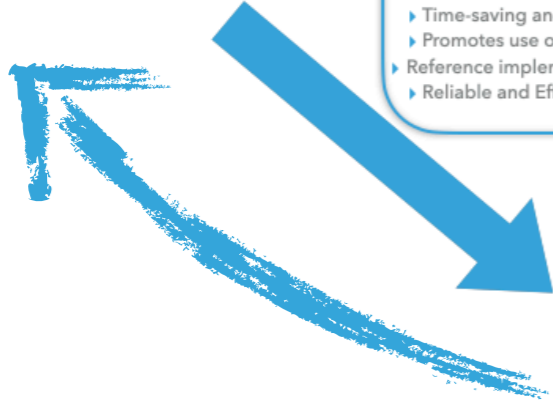


[newstalkzb.co.nz/news/education/modern-lego-sets-more-complex-less-inspiring/](http://newstalkzb.co.nz/news/education/modern-lego-sets-more-complex-less-inspiring/)

Query   Statistical confidence  $\alpha, \delta$

**Statistical Model Checking**

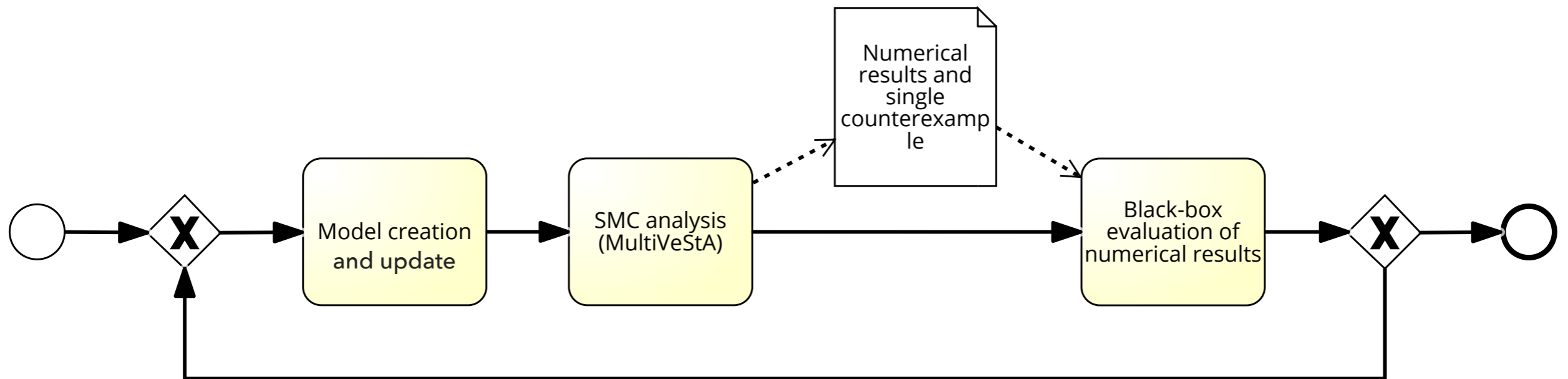
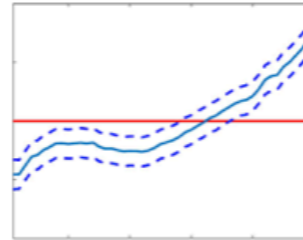
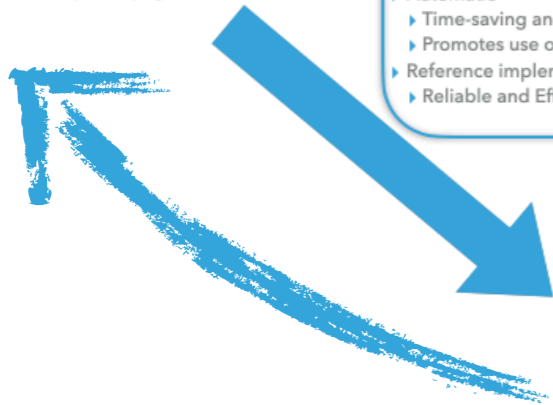
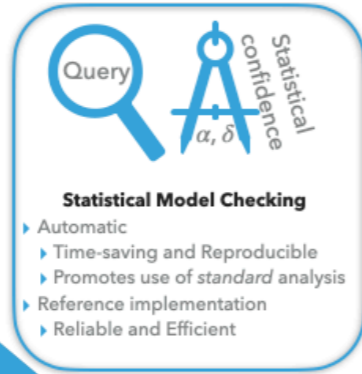
- ▶ Automatic
- ▶ Time-saving and Reproducible
- ▶ Promotes use of *standard* analysis
- ▶ Reference implementation
- ▶ Reliable and Efficient



# OUR METHODOLOGY 4 WHITE-BOX VALIDATION



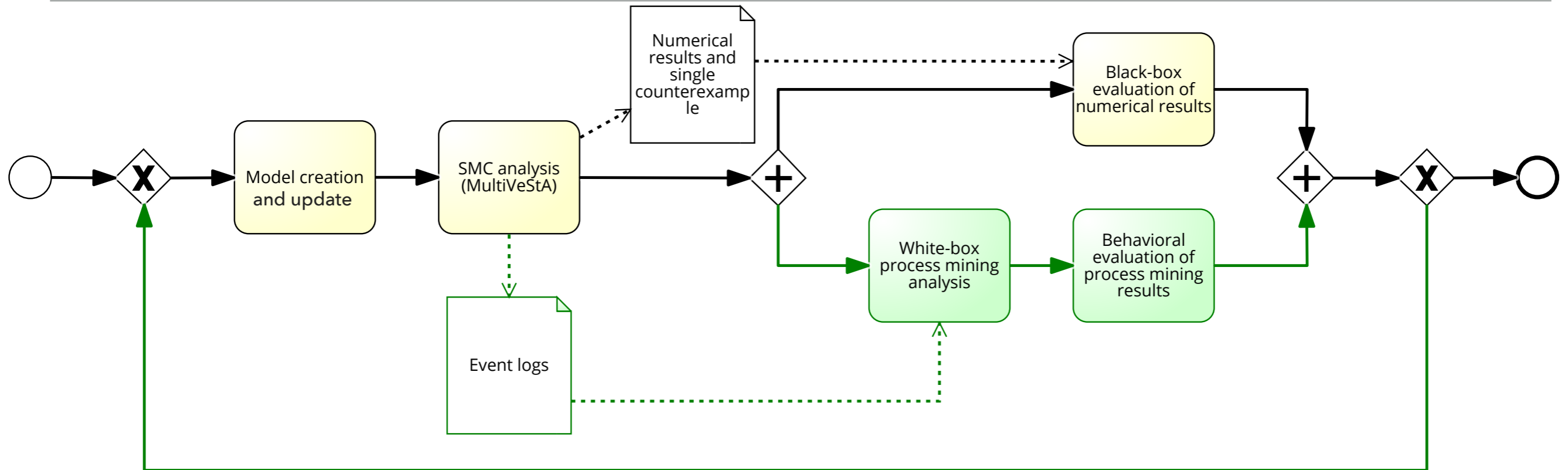
[newstalkzb.co.nz/news/education/modern-lego-sets-more-complex-less-inspiring/](http://newstalkzb.co.nz/news/education/modern-lego-sets-more-complex-less-inspiring/)



Informed guess driven by numerical results

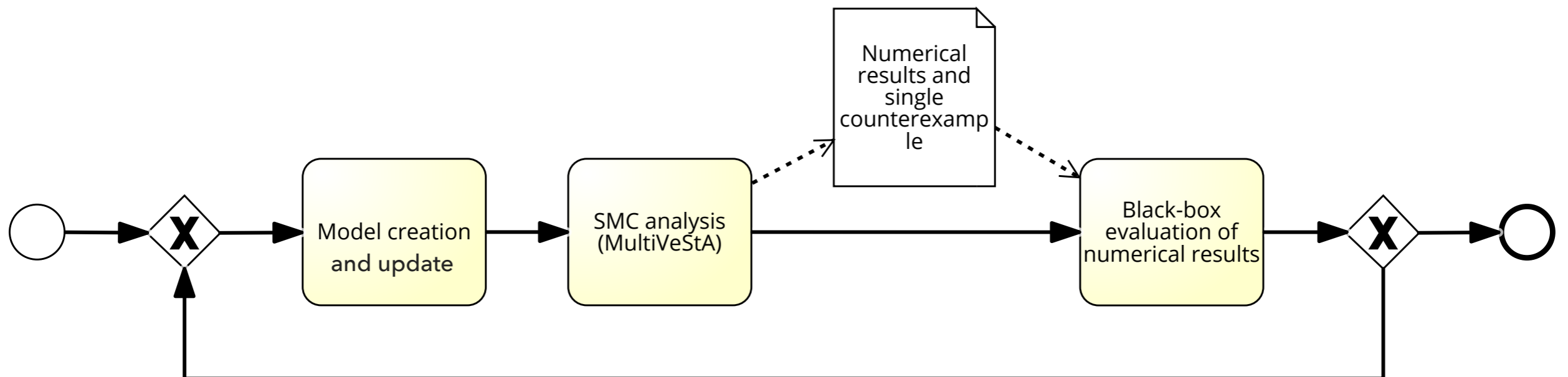
State-of-the-art life-cycle of SMC-analysed simulation

# OUR METHODOLOGY 4 WHITE-BOX VALIDATION



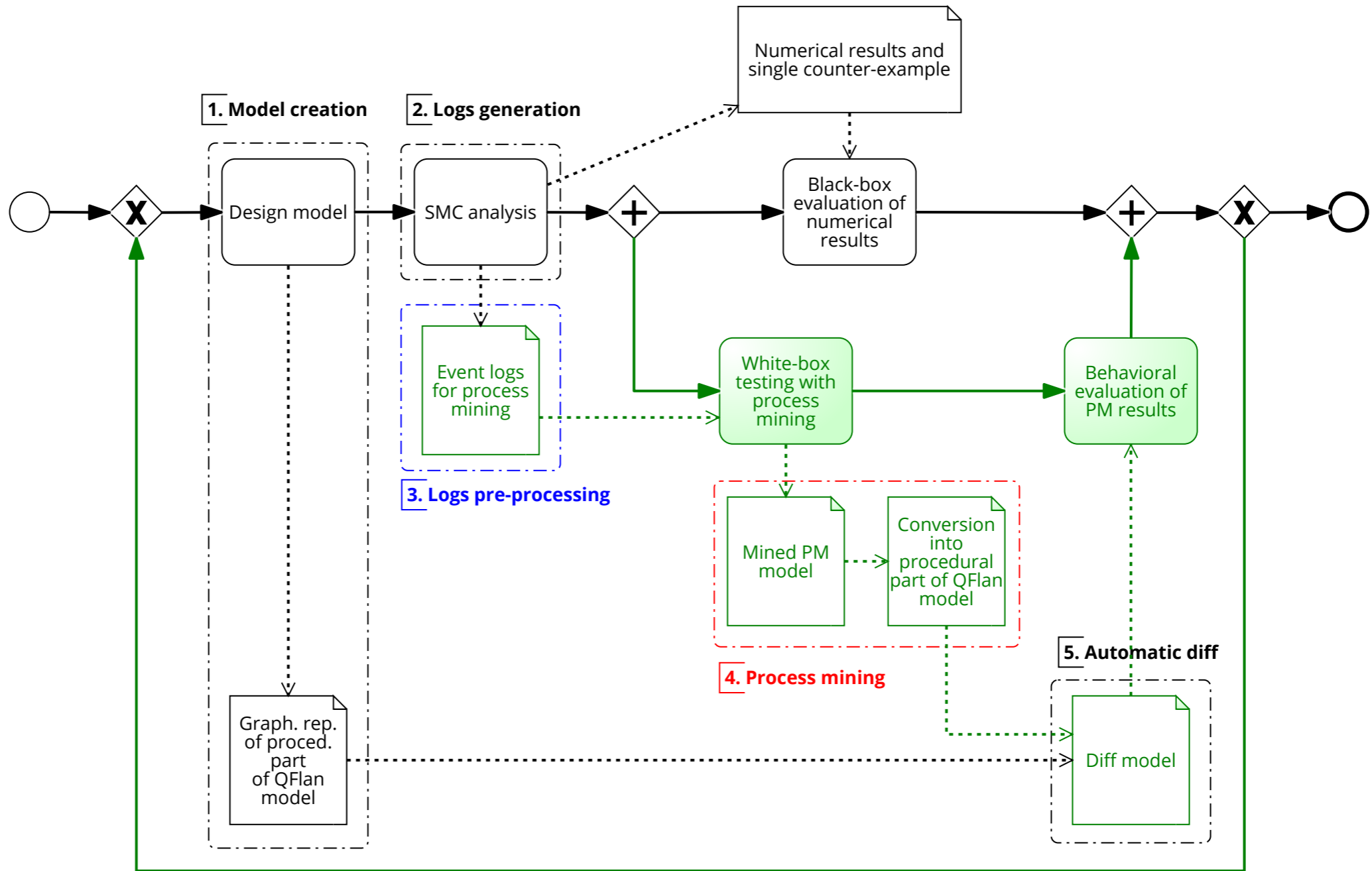
Unexpected behavior discovered with process mining and numerical results

Our novel SMC- and PM-guided methodology for white-box model validation



Informed guess driven by numerical results

Usual life-cycle of SMC-analysed simulation models



Unexpected behavior discovered with process mining and numerical results

# APPLICATION ON SIMPLE THREAT ANALYSIS EXAMPLE

The screenshot displays the RisQFLan application interface with several key components highlighted by blue boxes:

- Project Explorer:** Shows a project structure with folders like BGP, Bypassing802, CatBurglar, company\_damage, COMPANY\_DAMAGE, JuvenileDelinquent, RobBank, RobBank2, and SMC. The SMC folder contains MultiVeStA\_OUTPUT and src-gen.
- RisQFLan Editor:** Contains a code editor with the following snippet:
 

```

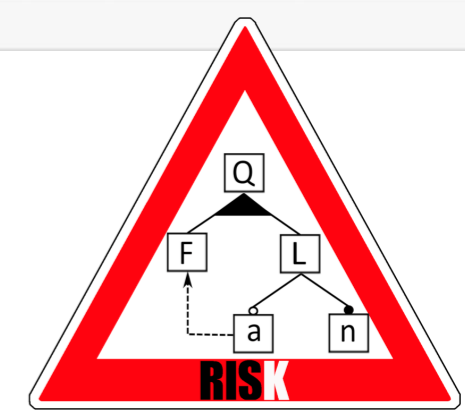
begin model RobBank_one_child
  begin variables
    AttackAttempts = 0
  end variables
  begin attack nodes
    RobBank
    OpenVault
    BlowUp
  end attack nodes
  begin defense nodes
  end defense nodes
  begin countermeasure nodes
    LockDown = {BlowUp}
  end countermeasure nodes
  begin attack diagram
    RobBank -> {OpenVault,BlowUp}
    RobBank -> {LockDown}
  end attack diagram
  begin attackers
    Thief
  end attackers
end model
      
```
- Console View:** Displays simulation output text, including "MultiVeStA client: sta" and "alpha: 0.1".
- Model Visualizer:** Shows a hierarchical tree diagram with nodes: RobBank (parent), OpenVault and BlowUp (children), and LockDown (grandchild). BlowUp has attributes: Cost = 90.0, Detection Rate = 0.0. LockDown has attribute: Defense Effectiveness A1.1 : RobBank = 0.1.
- Outline View:** Lists the model's structure: 1 Variable, 3 Attack Nodes, 0 Defense Node, 1 Countermeasure Node, Attack Tree, 1 Attacker, Defense Effectiveness, 1 Attack Detection Rate, 1 Attribute, and 1 Quantitative Constraint.
- Plot View:** Shows a line graph titled "MultiVeStA analysis of RobBank\_one\_child.bbt SMC of queryRobBank\_one\_child.multiquatex." with a y-axis ranging from 0.0 to 0.75.

Computers & Security  
Volume 109, October 2021, 102381

TC 11 Briefing Papers

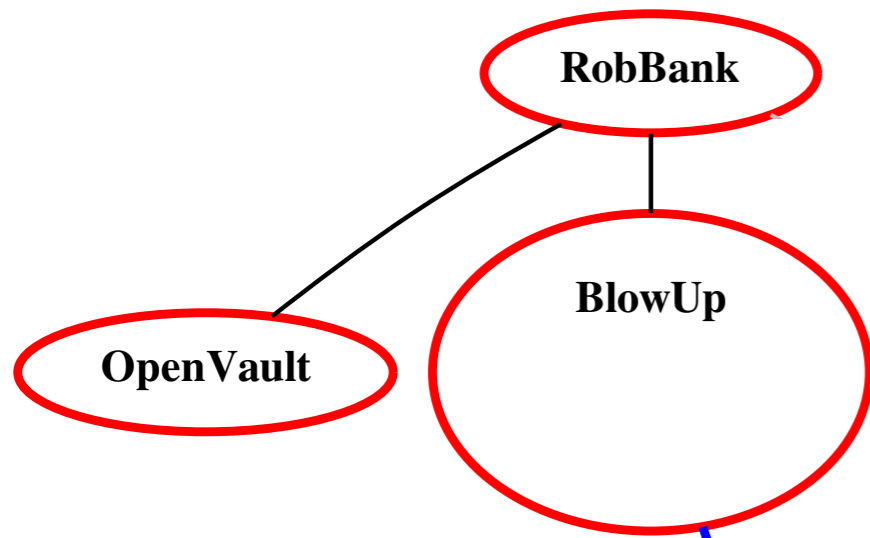
## Quantitative Security Risk Modeling and Analysis with RisQFLan

Maurice H. ter Beek <sup>a,\*,</sup>, Axel Legay <sup>b,</sup>, Alberto Lluch Lafuente <sup>c,</sup>, Andrea Vandin <sup>c, d</sup>

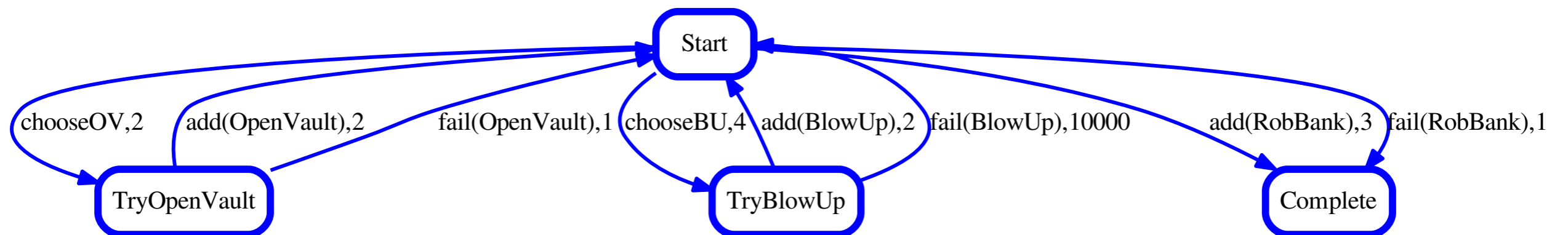
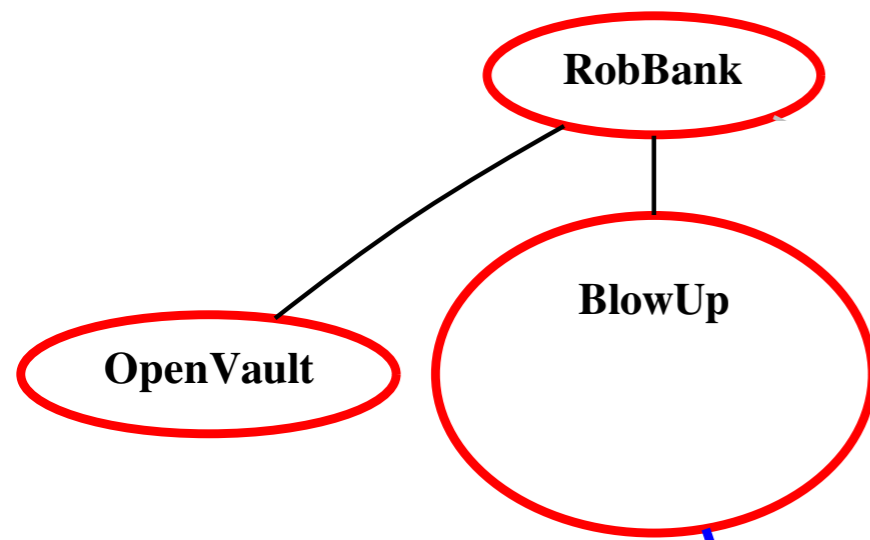


# APPLICATION ON SIMPLE THREAT ANALYSIS EXAMPLE

---

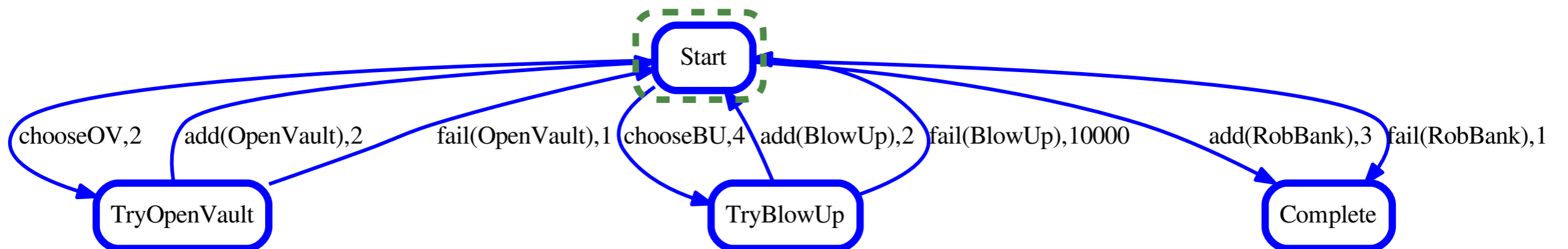
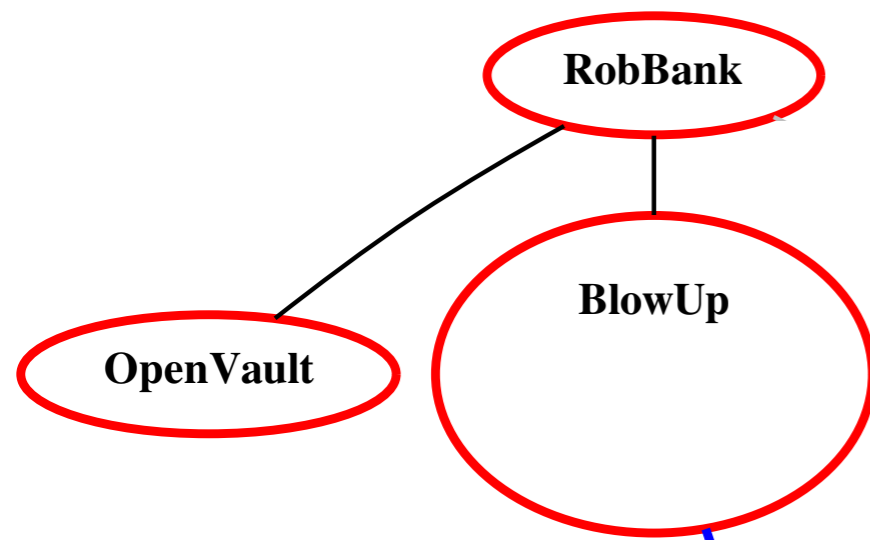


# APPLICATION ON SIMPLE THREAT ANALYSIS EXAMPLE

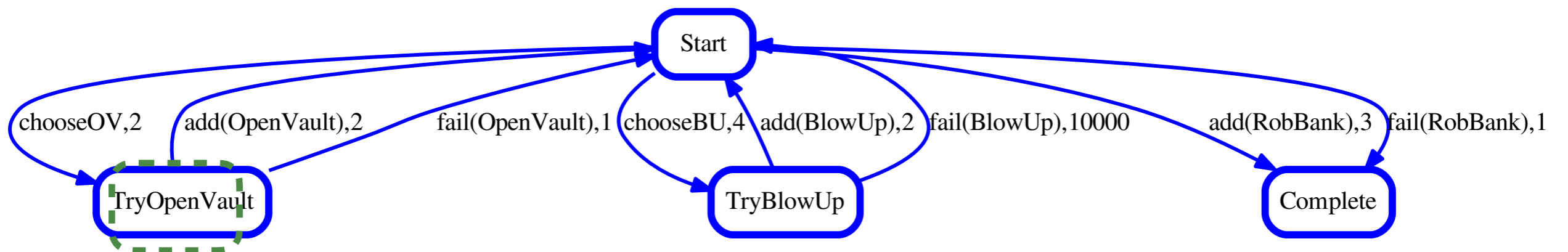
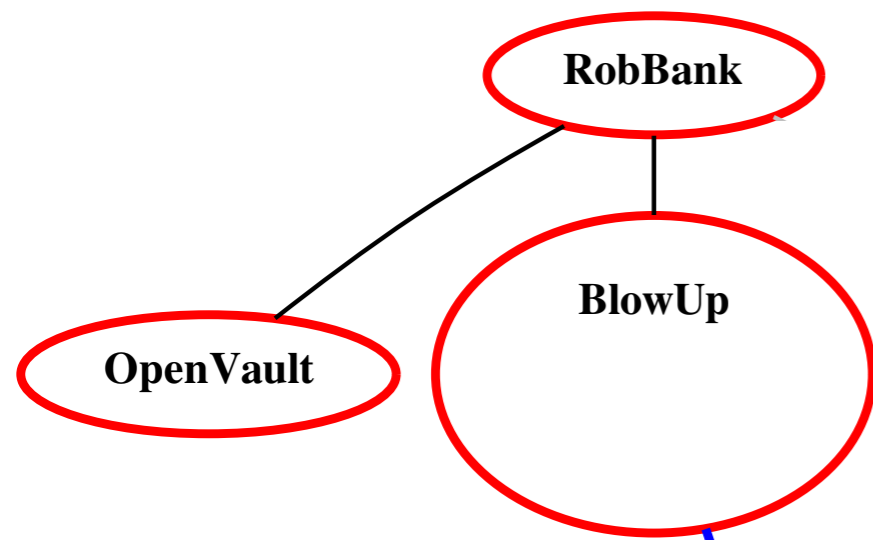




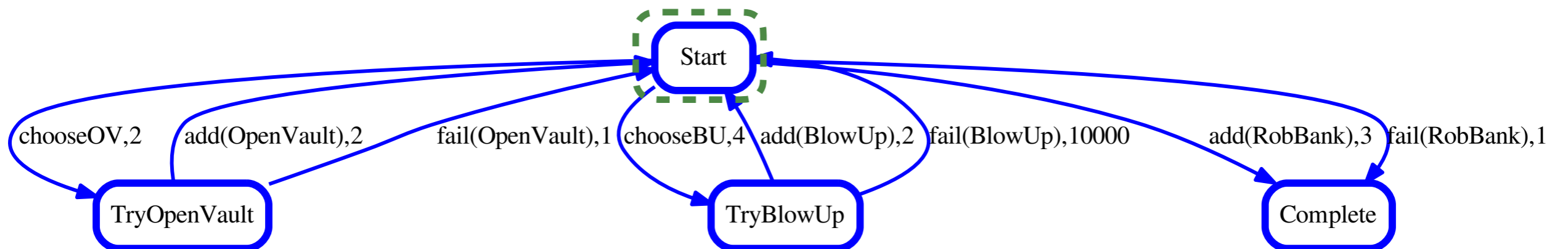
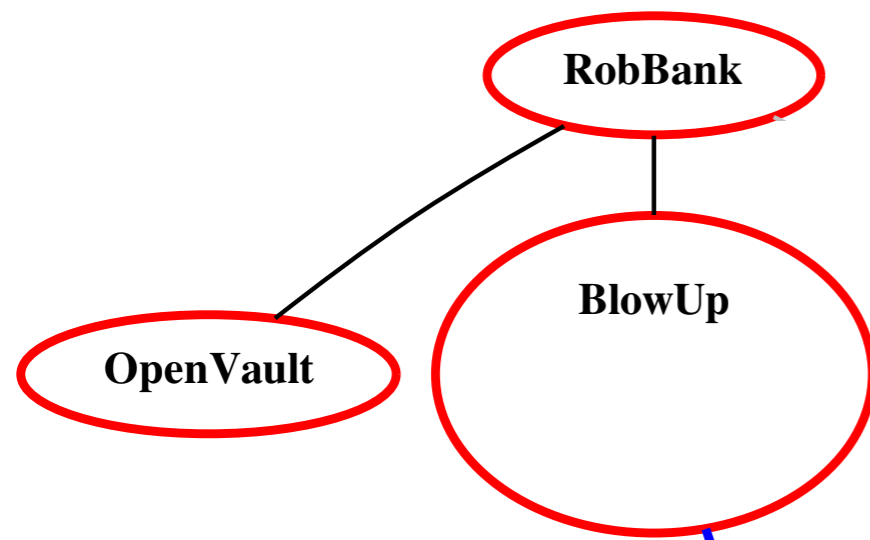
# APPLICATION ON SIMPLE THREAT ANALYSIS EXAMPLE



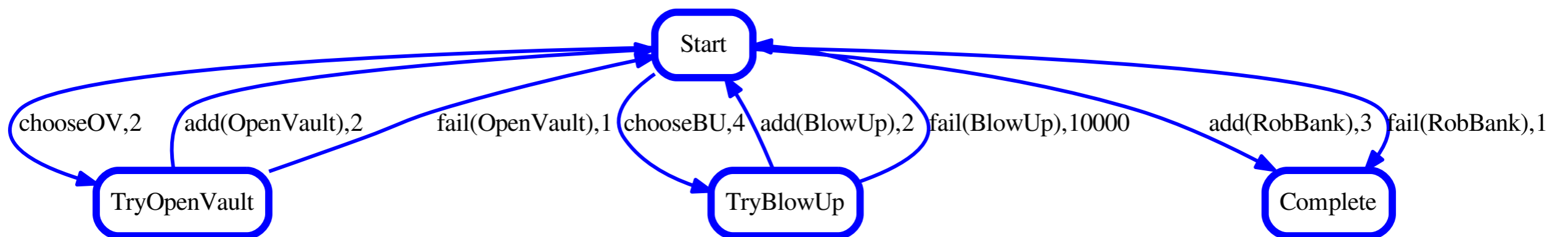
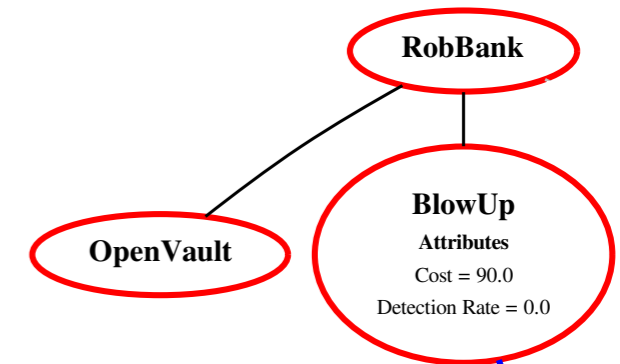
# APPLICATION ON SIMPLE THREAT ANALYSIS EXAMPLE



# APPLICATION ON SIMPLE THREAT ANALYSIS EXAMPLE



# ANALYSIS OF ORIGINAL MODEL

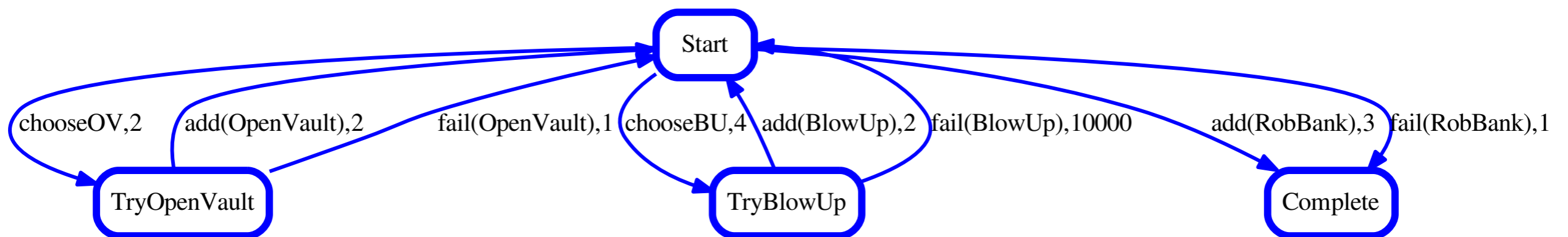
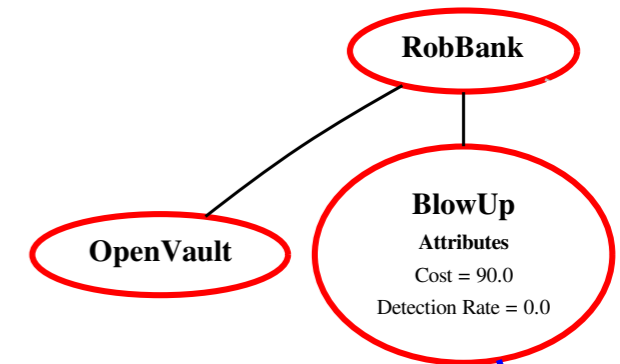


# ANALYSIS OF ORIGINAL MODEL

Probability of successful bank robbery!? **0.17**

**Why?**

- 1) My defences are *good*
- 2) The attacker is *bad*
- 3) **Or my model is bad!?**



# ANALYSIS OF ORIGINAL MODEL

Probability of successful bank robbery!? **0.17**

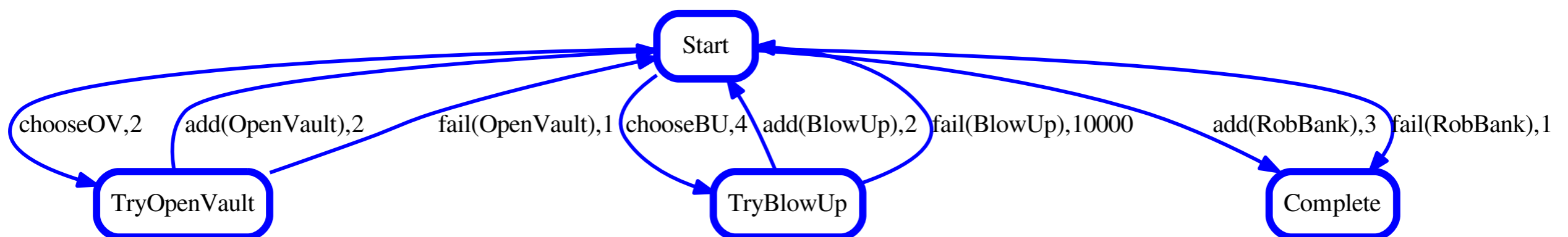
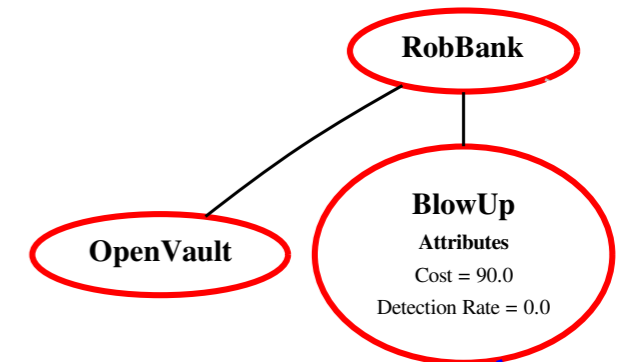
**Why?**

- 1) My defences are *good*
- 2) The attacker is *bad*
- 3) **Or my model is bad!?**

We set  $\alpha=0.1$ ,  $\delta=0.1$

MultiVeStA performs **240 simulations**

- We generate **logs for each simulation**
- We ask **Fluxicon Disco** mine these logs
- Can we spot **any issue in the model?**



# ANALYSIS OF ORIGINAL MODEL

Probability of successful bank robbery!? **0.17**

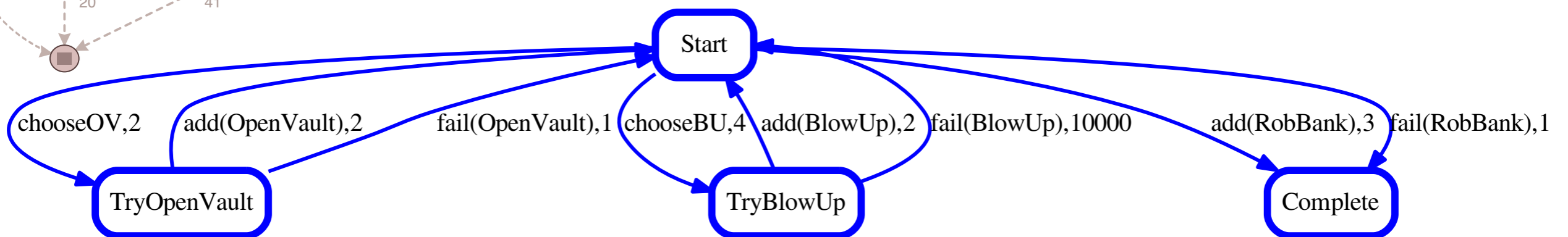
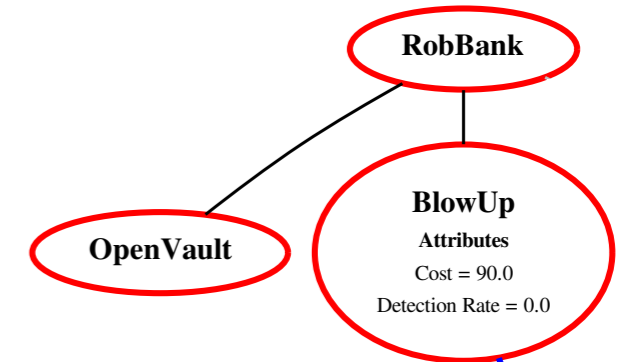
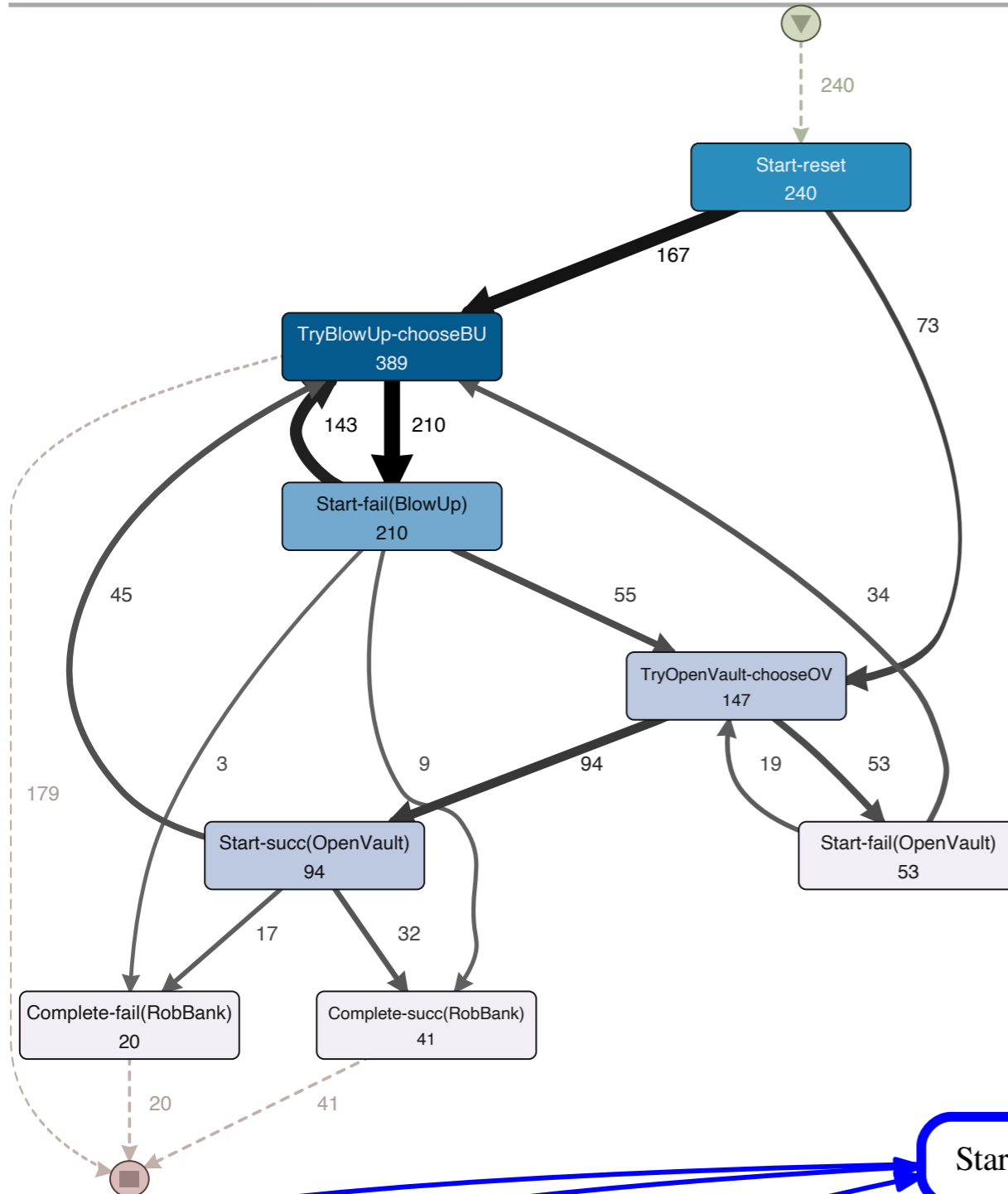
**Why?**

- 1) My defences are *good*
- 2) The attacker is *bad*
- 3) **Or my model is bad!?**

We set  $\alpha=0.1$ ,  $\delta=0.1$

MultiVeStA performs **240 simulations**

- ▶ We generate **logs** for each simulation
- ▶ We ask **Fluxicon Disco** mine these logs
- ▶ Can we spot **any issue in the model?**



# ANALYSIS OF ORIGINAL MODEL

Probability of successful bank robbery!? **0.17**

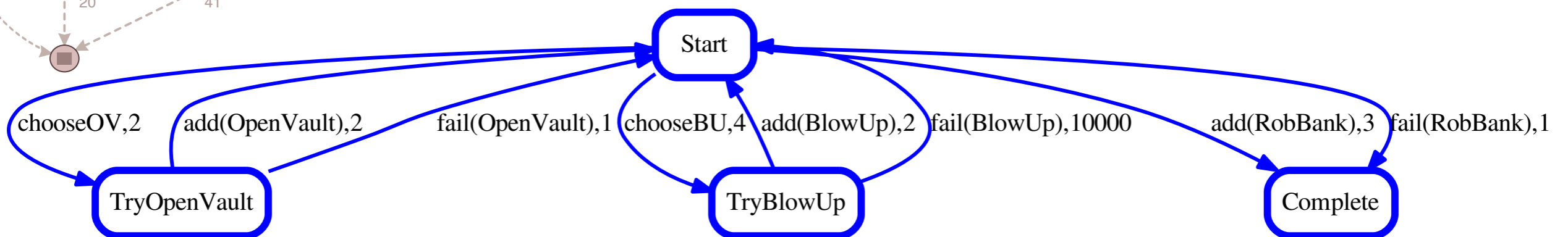
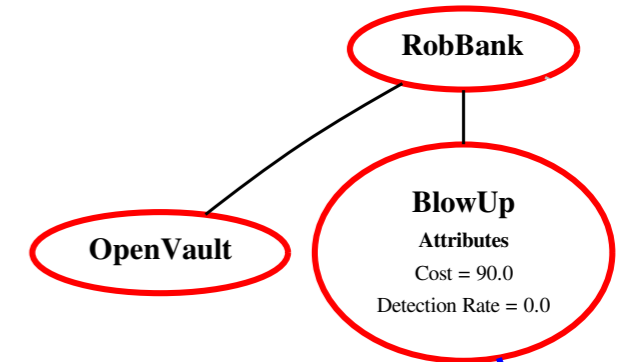
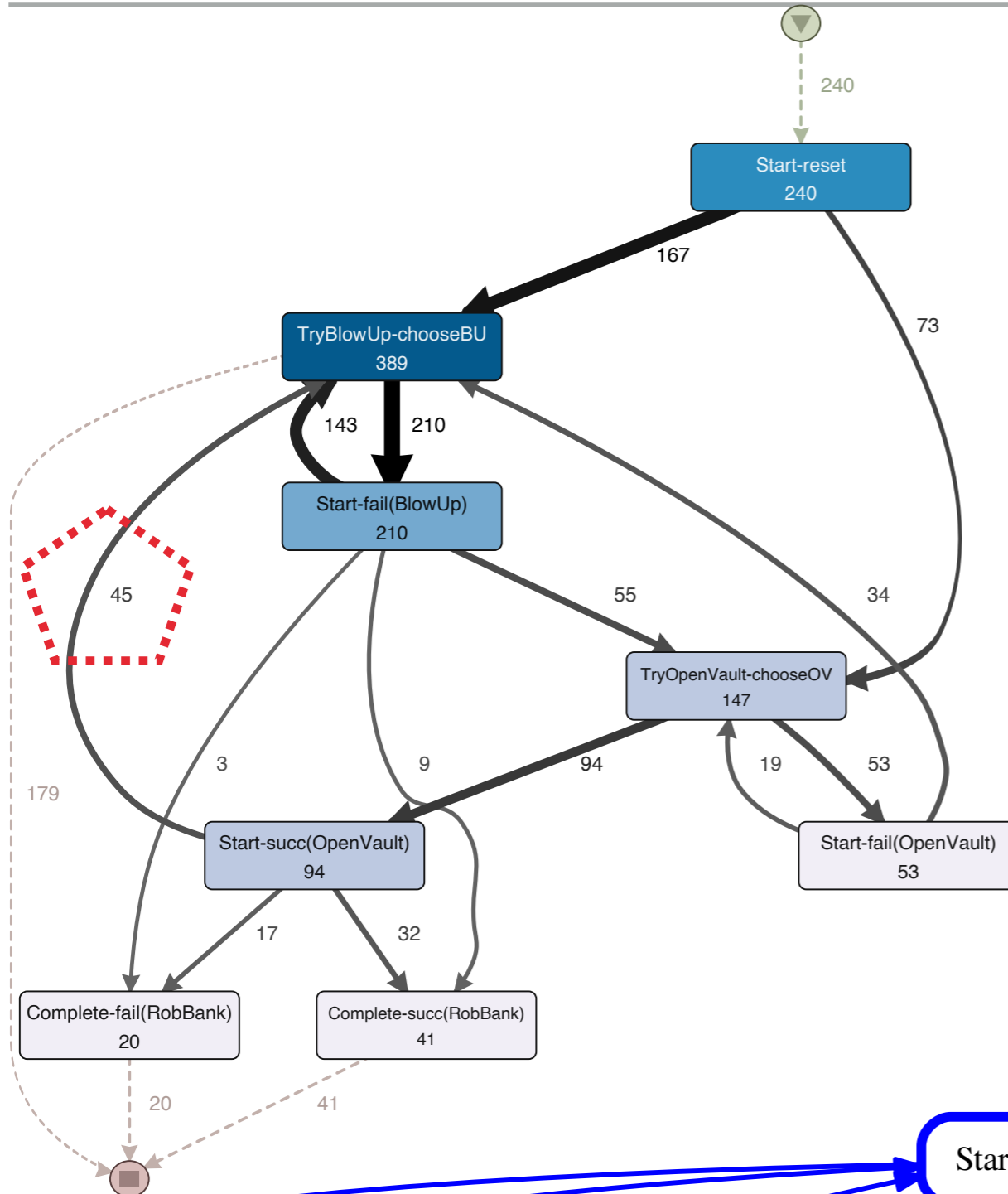
**Why?**

- 1) My defences are *good*
- 2) The attacker is *bad*
- 3) **Or my model is bad!?**

We set  $\alpha=0.1$ ,  $\delta=0.1$

MultiVeStA performs **240 simulations**

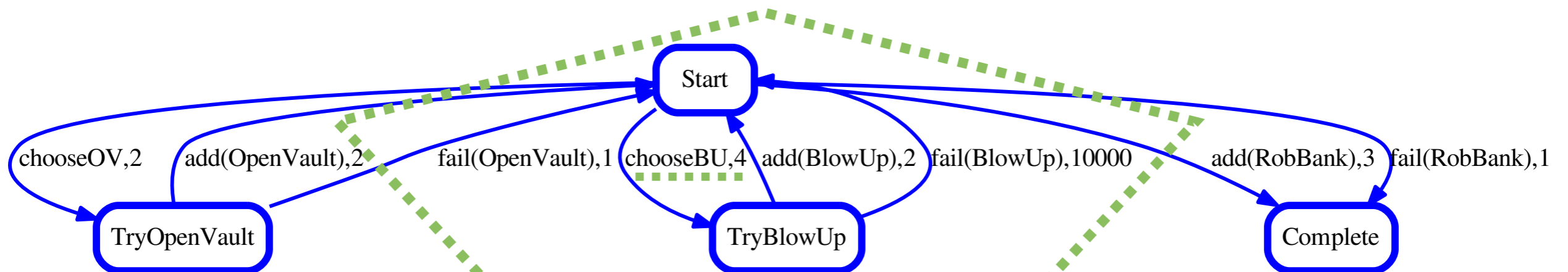
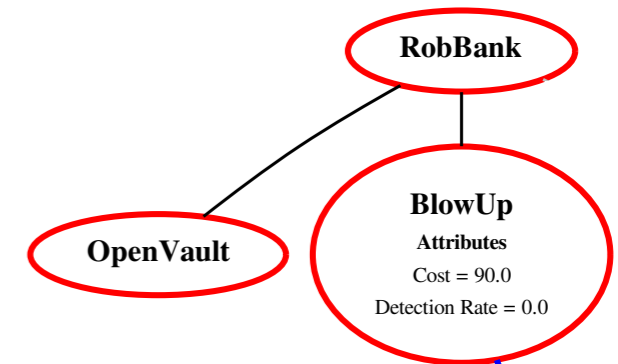
- ▶ We generate **logs** for each simulation
- ▶ We ask **Fluxicon Disco** mine these logs
- ▶ Can we spot **any issue in the model?**





# FIRST REFINEMENT: PARSIMONIOUS ATTACKER

```
1 // We add the !allowed(RobBank)
2 Start -(chooseBU, 4, !allowed(RobBank)) -> TryBlowUp,
3 TryBlowUp -(succ(BlowUp), 2) -> Start,
4 TryBlowUp -(fail(BlowUp), 10000) -> Start
```



# FIRST REFINEMENT: PARSIMONIOUS ATTACKER – ANALYSIS

Probability of successful bank robbery!? ~~0.17~~ 0.31

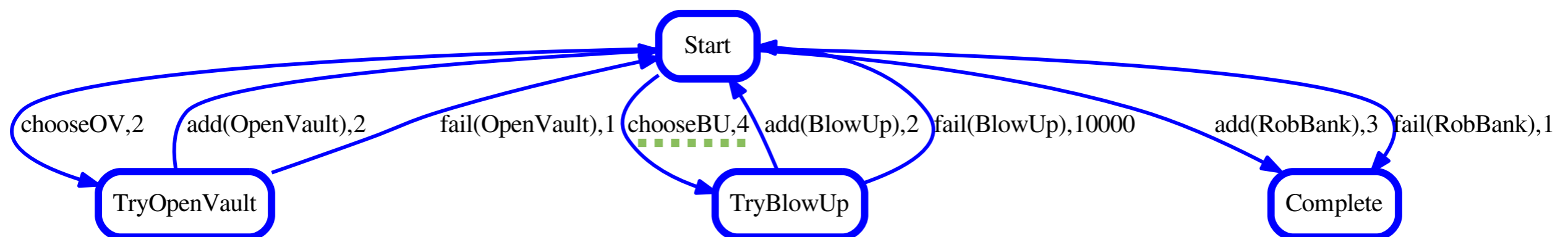
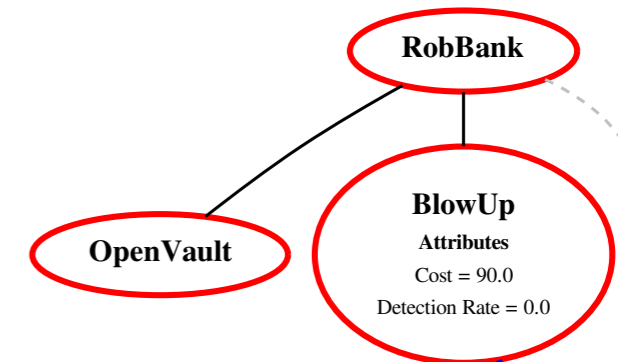
**Why?**

- 1) My defences are *good*
- 2) The attacker is *bad*
- 3) **Or my model is bad!?**

We set  $\alpha=0.1$ ,  $\delta=0.1$

MultiVeStA performs **240 simulations**

- We generate **logs for each simulation**
- We ask **Fluxicon Disco** mine these logs
- Can we spot **any issue in the model?**



# FIRST REFINEMENT: PARSIMONIOUS ATTACKER – ANALYSIS

Probability of successful bank robbery!? ~~0.17~~ 0.31

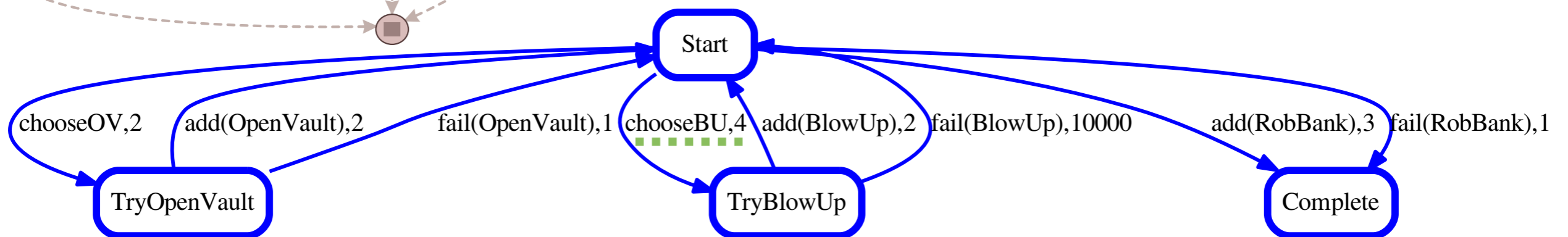
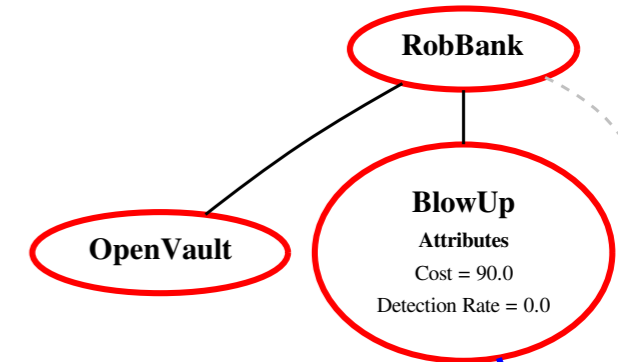
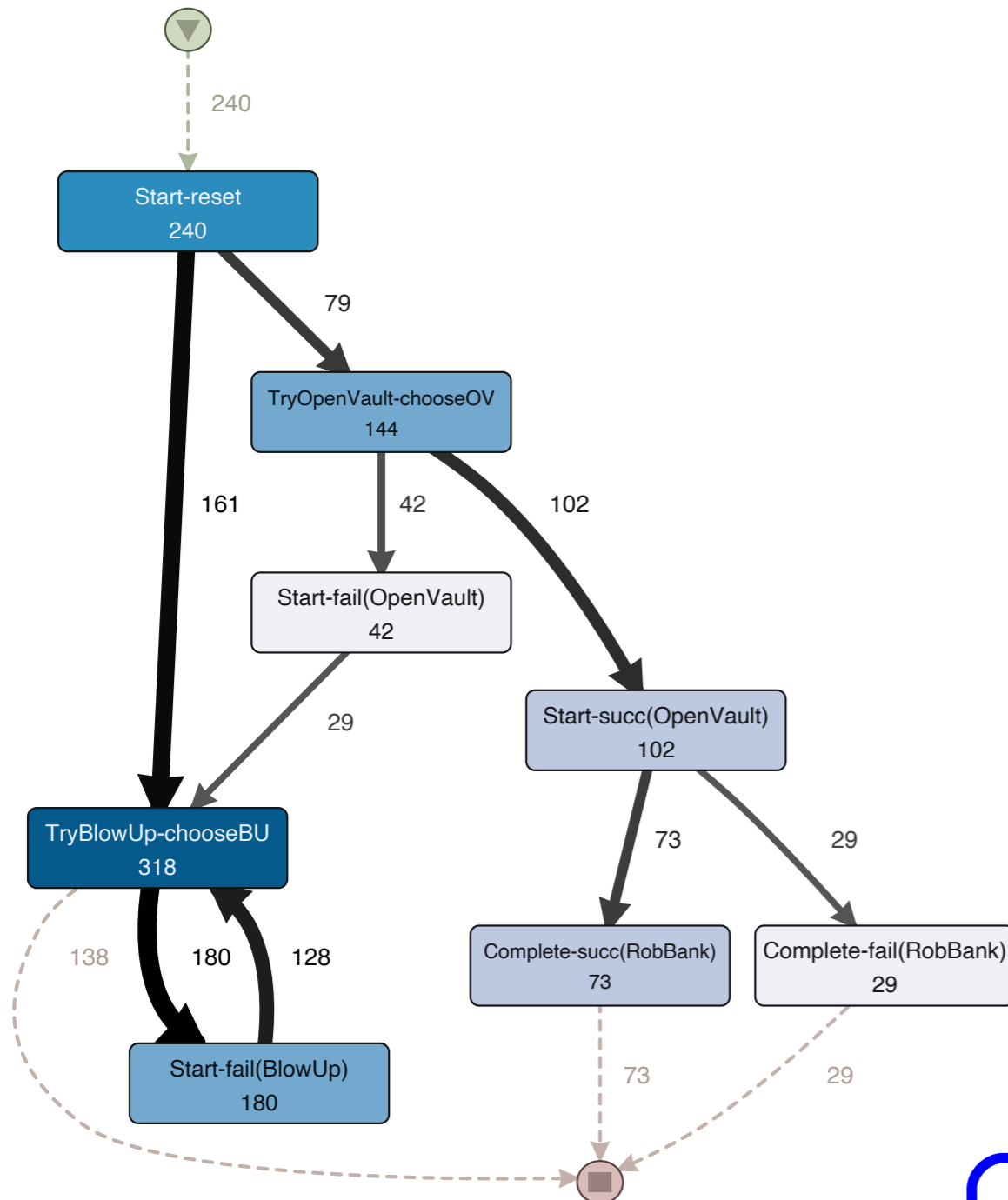
**Why?**

- 1) My defences are *good*
- 2) The attacker is *bad*
- 3) **Or my model is bad!?**

We set  $\alpha=0.1$ ,  $\delta=0.1$

MultiVeStA performs **240 simulations**

- We generate **logs** for each simulation
- We ask **Fluxicon Disco** mine these logs
- Can we spot **any issue in the model?**



# FIRST REFINEMENT: PARSIMONIOUS ATTACKER - ANALYSIS

Probability of successful bank robbery!? ~~0.17~~ 0.31

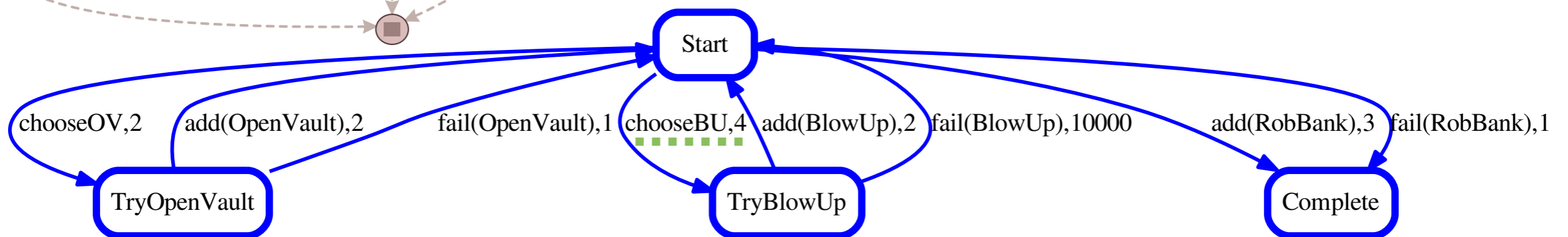
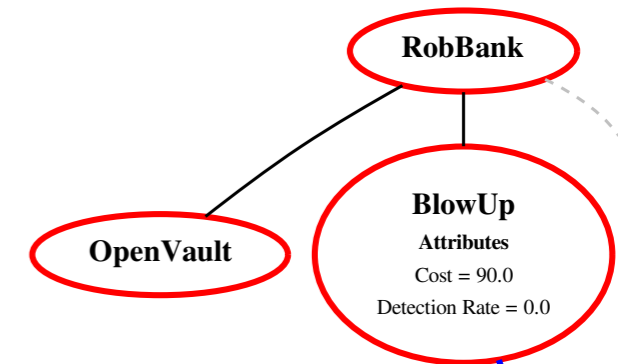
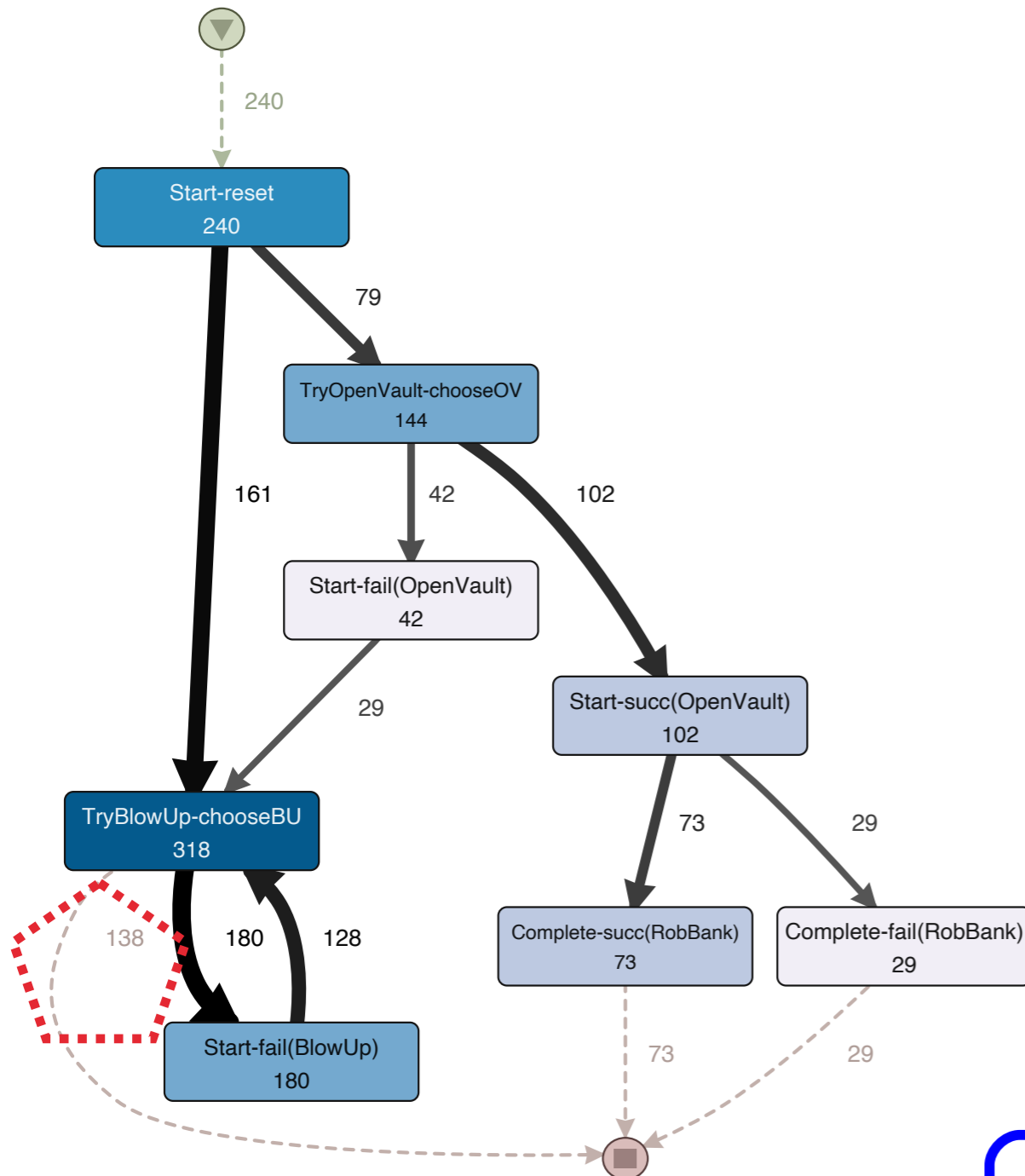
**Why?**

- 1) My defences are *good*
- 2) The attacker is *bad*
- 3) **Or my model is bad!?**

We set alpha=0.1, delta=0.1

MultiVeStA performs **240 simulations**

- We generate **logs** for each simulation
- We ask **Fluxicon Disco** mine these logs
- Can we spot **any issue in the model?**

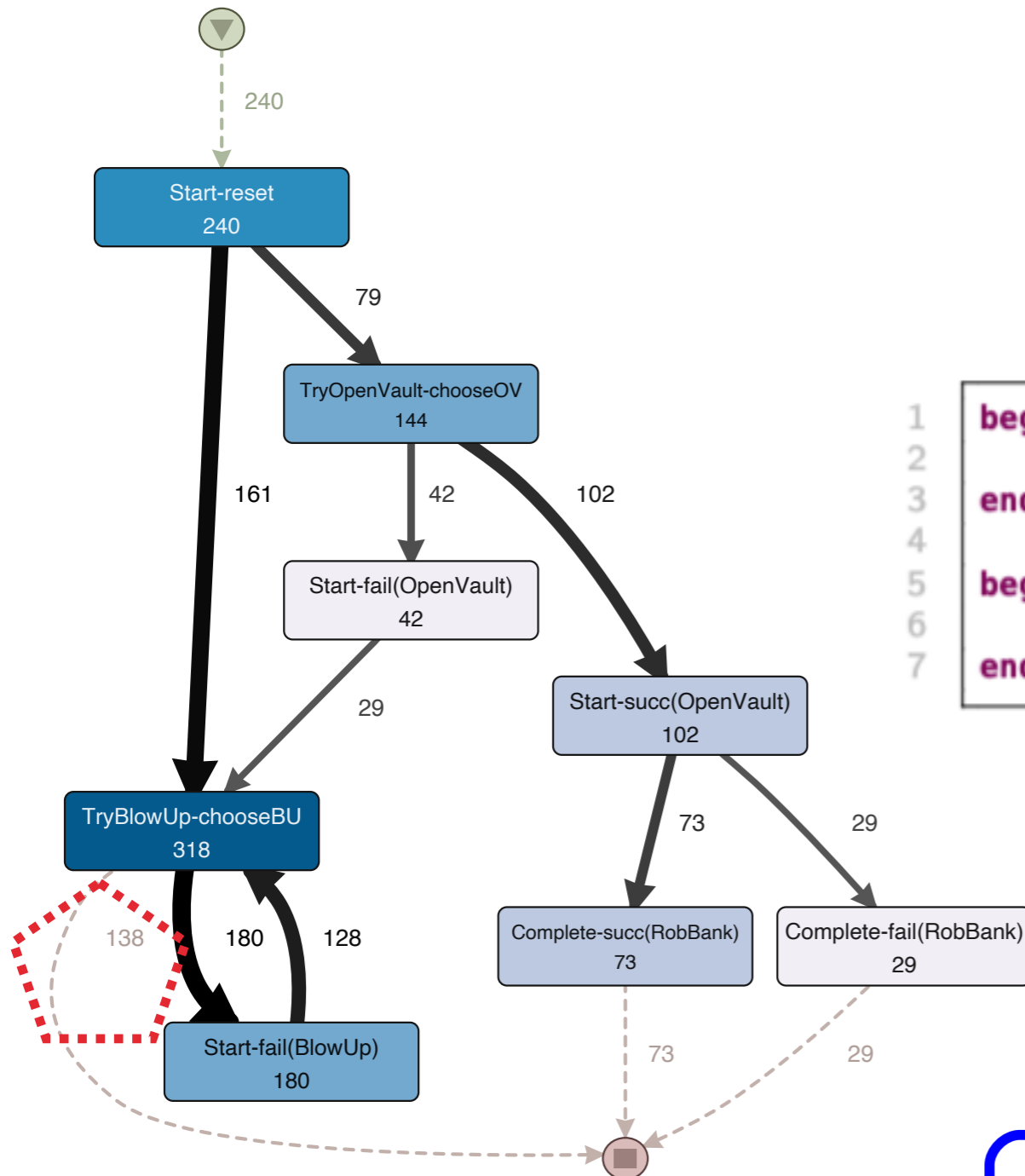


# FIRST REFINEMENT: PARSIMONIOUS ATTACKER - ANALYSIS

Probability of successful bank robbery!? ~~0.17~~ 0.31

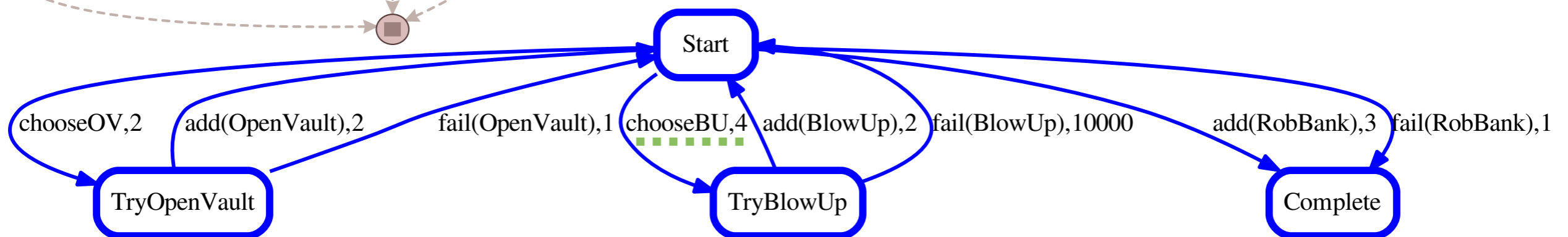
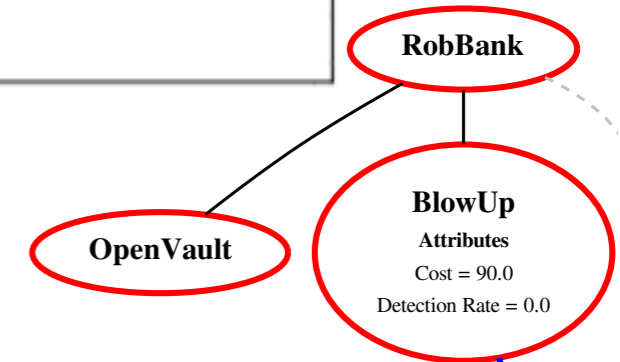
Why?

- 1) My defences are *good*
- 2) The attacker is *bad*
- 3) **Or my model is bad!?**



```

1 begin attributes
2   Cost = {BlowUp = 90, OpenVault = 0}
3 end attributes
4
5 begin quantitative constraints
6   { value(Cost) <= 100 }
7 end quantitative constraints
    
```



# SECOND REFINEMENT

```

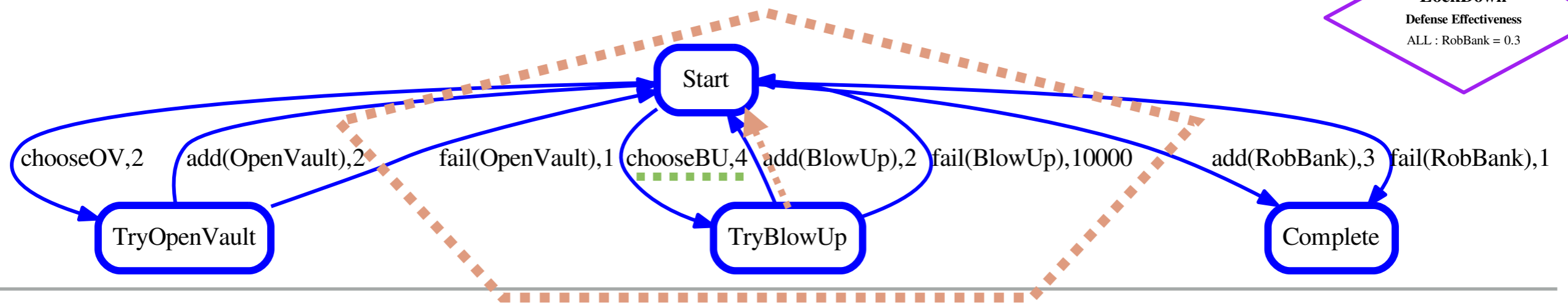
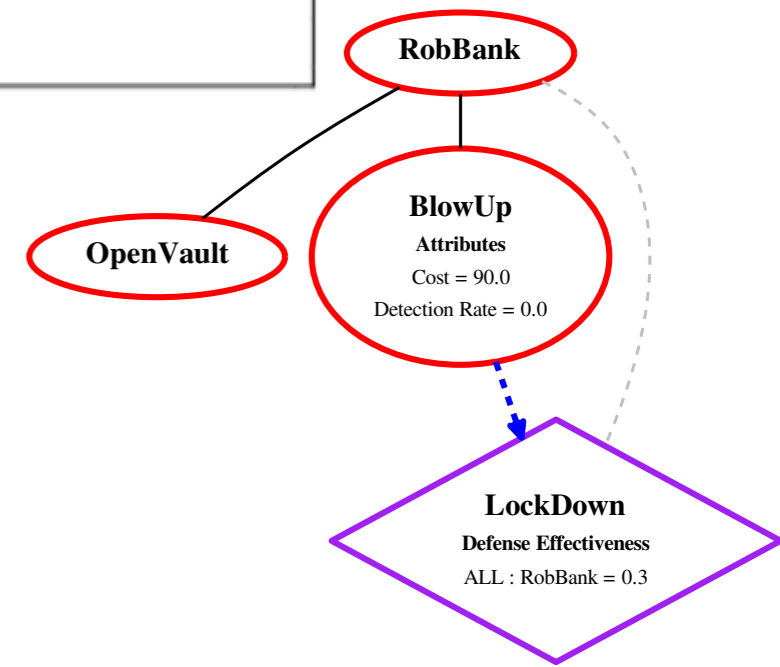
1 begin actions
2   chooseOV
3   chooseBU
4   goBack
5 end actions
  
```

```

1 begin attributes
2   Cost = {BlowUp = 90, OpenVault = 0}
3 end attributes
4
5 begin quantitative constraints
6   { value(Cost) <= 100 }
7 end quantitative constraints
  
```

```

1 // Strategy where the attacker tries to blow up the vault
2 Start -(chooseBU, 4, !allowed(RobBank)) -> TryBlowUp,
3 TryBlowUp -(succ(BlowUp), 2) -> Start,
4 TryBlowUp -(fail(BlowUp), 10000) -> Start,
5 TryBlowUp -(goBack, 0.00001) -> Start
  
```



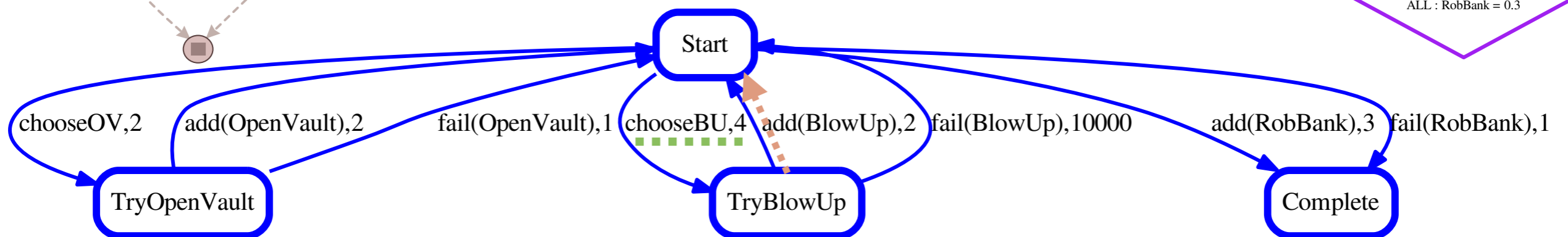
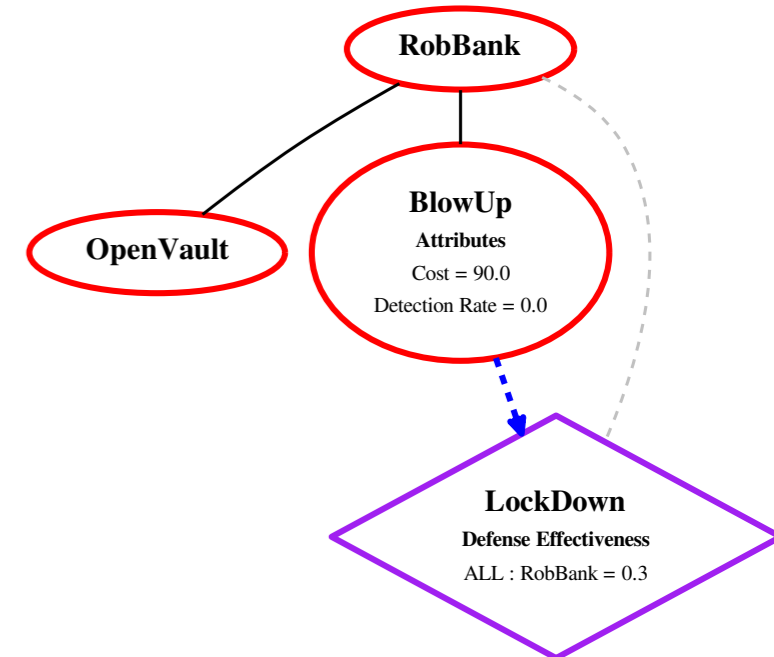
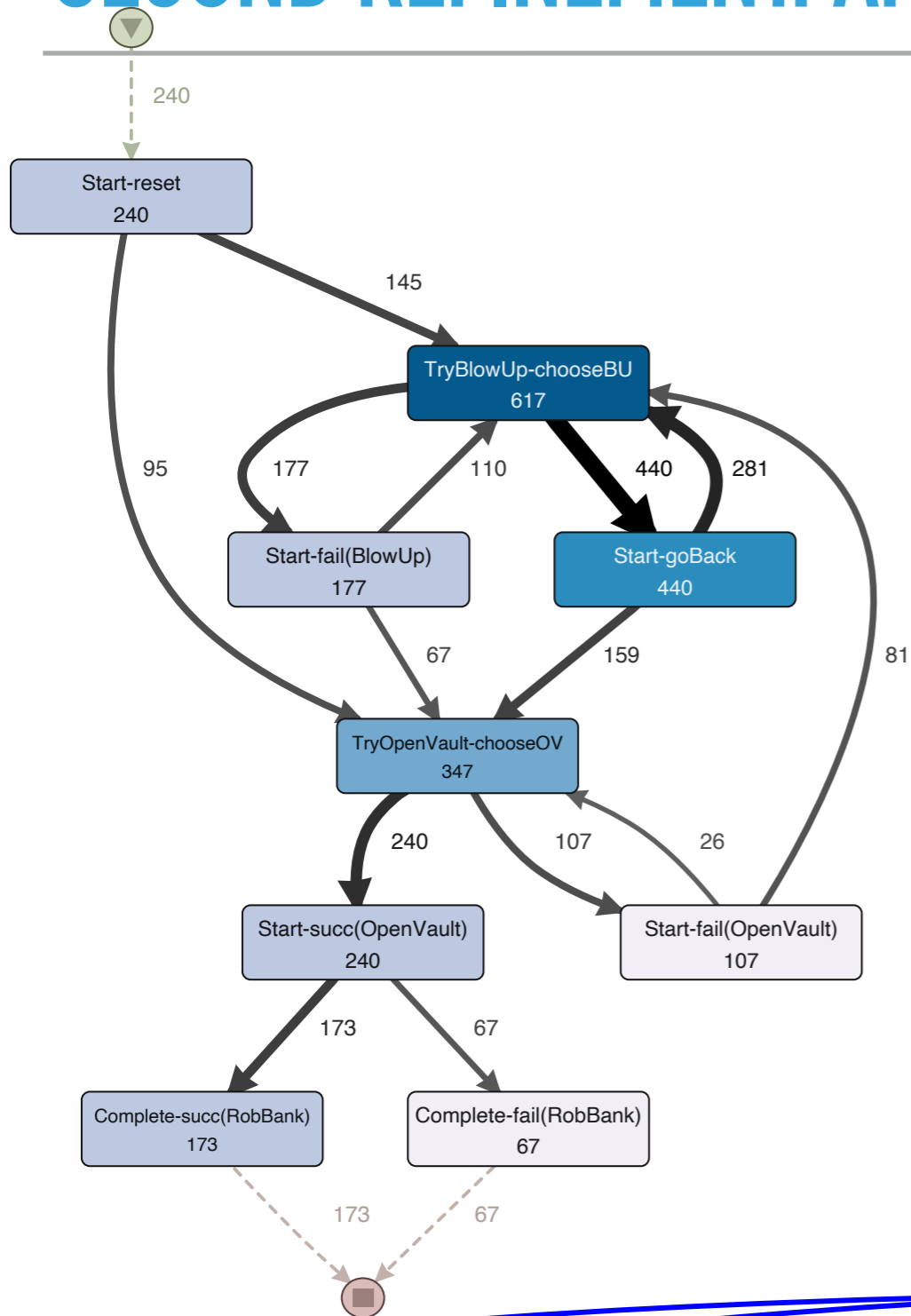
# SECOND REFINEMENT: ANALYSIS

Probability of successful bank robbery!? ~~0.17~~ ~~0.31~~ **0.72**

We set  $\alpha=0.1$ ,  $\delta=0.1$

MultiVeStA performs **240 simulations**

- ▶ We generate **logs** for each simulation
- ▶ We ask **Fluxicon Disco** mine these logs
- ▶ Can we spot **any issue** in the model?



# PM MEETS SMC: CONCLUSIONS & FUTURE WORKS

---

- ▶ We proposed a novel methodology for validating and enhancing simulation models to make them more reliable
  - ▶ We obtained: **SMC- and PM-guided white-box behavioral model validation and enhancement**
- ▶ **Future works**
  - ▶ More realistic models, from more domains (e.g., ABM from social sciences)
  - ▶ Conformance checking might help our white-box analysis
  - ▶ Currently, we use **PM after SMC**:
    - ▶ **Using PM during SMC**: streaming PM might help improving SMC analysis
    - ▶ **Using PM before SMC**: discovery algorithms might be applied to real data to
      - ▶ synthesize attack-defense trees and/or attacker behaviors
      - ▶ or parts of simulation models in general



THANK YOU FOR  
YOUR ATTENTION!

---

QUESTIONS?  
FEEDBACK?

---

Andrea Vandin  
[andrea.vandin@santannapisa.it](mailto:andrea.vandin@santannapisa.it)  
[www.santannapisa.it/en/andrea-vandin](http://www.santannapisa.it/en/andrea-vandin)