# Synthesizing Intensional Behavior Models by Graph Transformation

Carlo Ghezzi,
*Andrea Mocci,*
Mattia Monga

deep se dependable evolvable pervasive software engineering

## A Problem

Analysis

Input — Specification

Specification Recovery

## Specification Recovery

? Analysis ?

Input — Specification

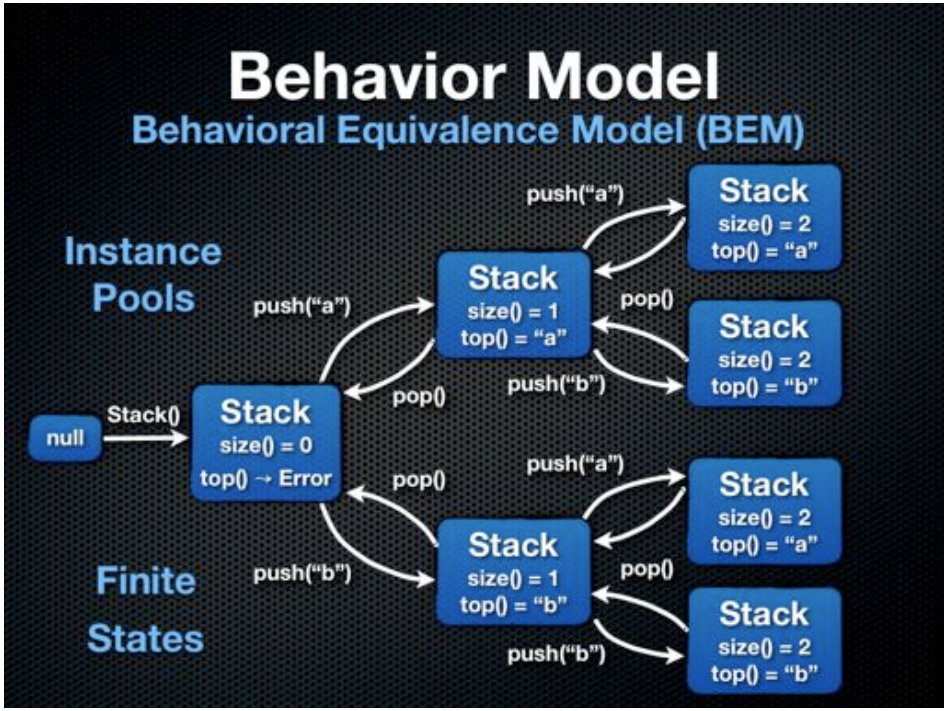
## Specification Recovery

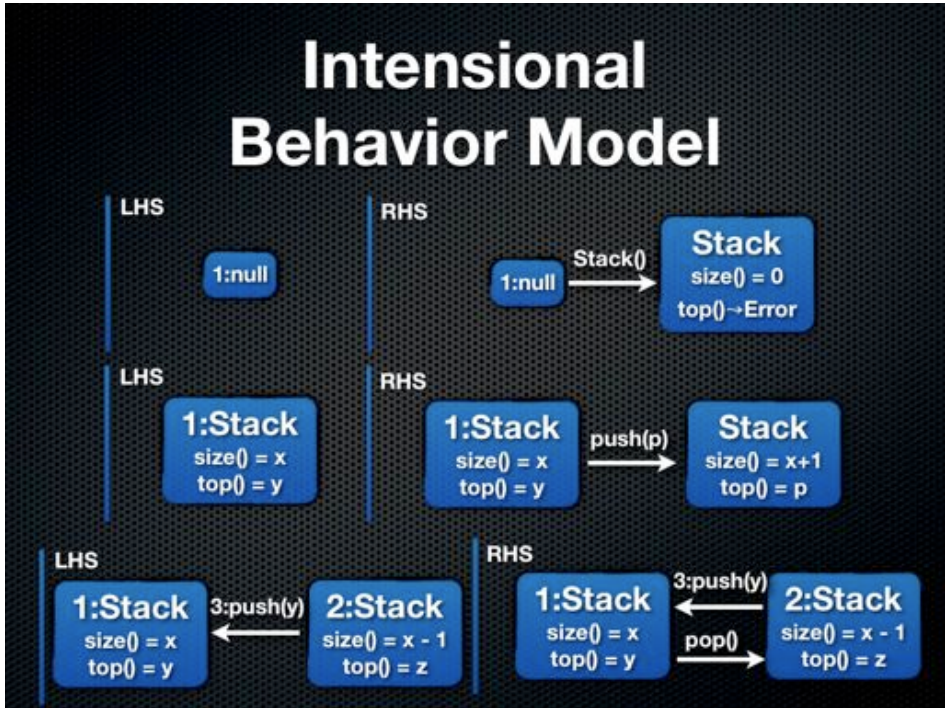
Black-Box Dynamic Analysis

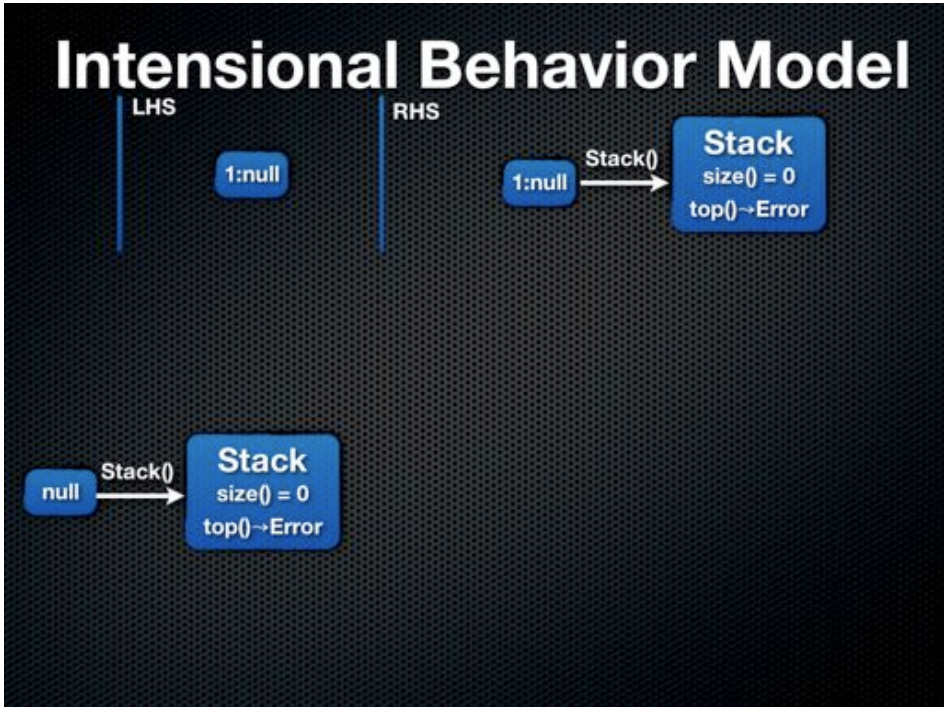
?

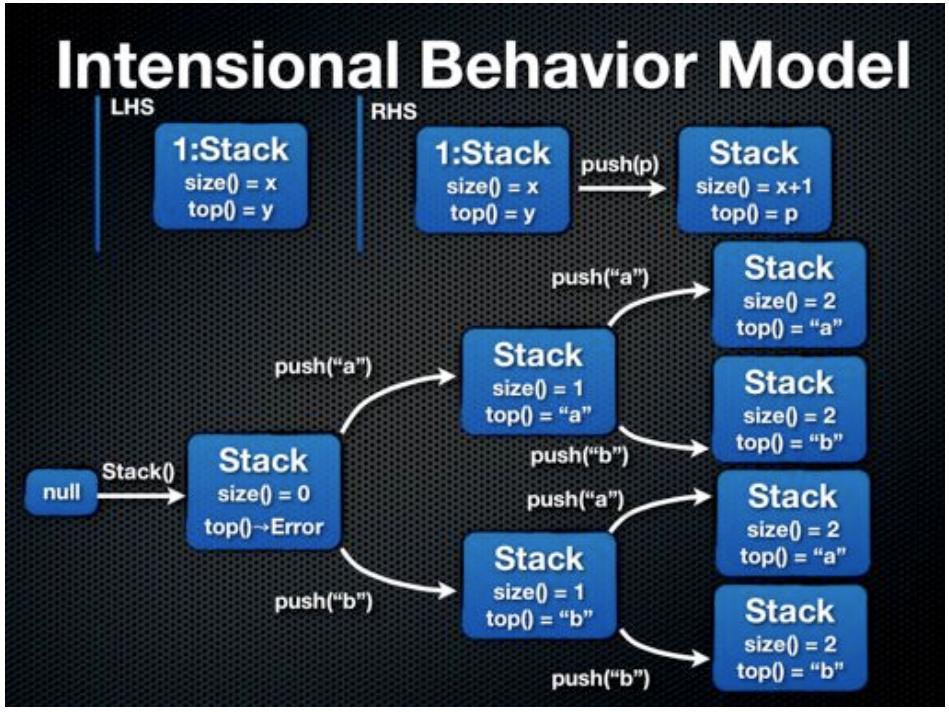
Data Abstractions — Intensional Behavior Models

Is it relevant?

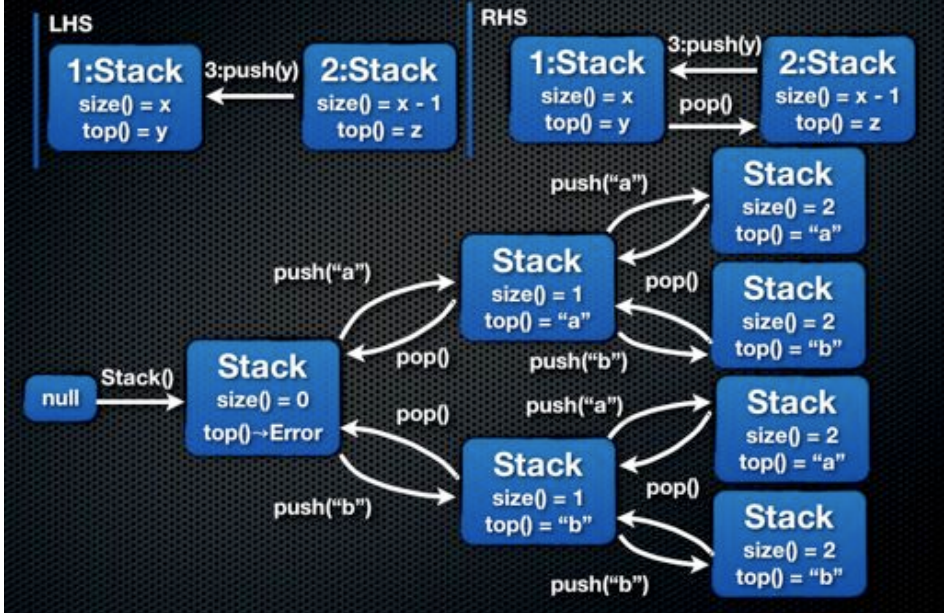

Behavior Model
Behavioral Equivalence Model (BEM)
Instance Pools
Finite States
null
Stack()
Stack
size() = 0
top() → Error
push("a")
pop()
push("b")
Stack
size() = 1
top() = "a"
Stack
size() = 2
top() = "a"
Stack
size() = 2
top() = "b"
Stack
size() = 1
top() = "b"
Stack
size() = 2
top() = "a"
Stack
size() = 2
top() = "b"


Intensional Behavior Model
LHS
RHS
1:null
1:null
Stack()
Stack
size() = 0
top()→Error
1:Stack
size() = x
top() = y
1:Stack
size() = x
top() = y
push(p)
Stack
size() = x+1
top() = p
1:Stack
size() = x
top() = y
3:push(y)
2:Stack
size() = x - 1
top() = z
1:Stack
size() = x
top() = y
3:push(y)
2:Stack
size() = x - 1
top() = z
pop()

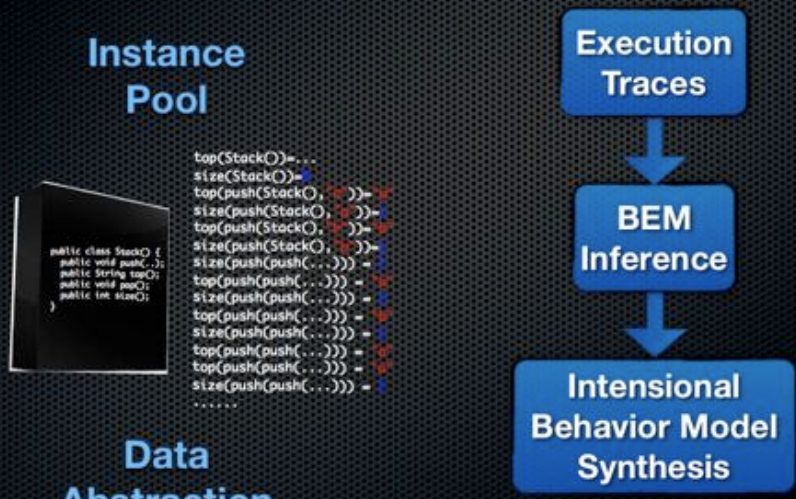

Intensional Behavior Model
LHS
RHS
1:null
1:null
Stack()
Stack
size() = 0
top()→Error
null
Stack()
Stack
size() = 0
top()→Error


Intensional Behavior Model
LHS
RHS
1:Stack
size() = x
top() = y
1:Stack
size() = x
top() = y
push(p)
Stack
size() = x+1
top() = p
null
Stack()
Stack
size() = 0
top()→Error
push("a")
push("b")
Stack
size() = 1
top() = "a"
Stack
size() = 1
top() = "b"
Stack
size() = 2
top() = "a"
Stack
size() = 2
top() = "b"
Stack
size() = 2
top() = "a"
Stack
size() = 2
top() = "b"

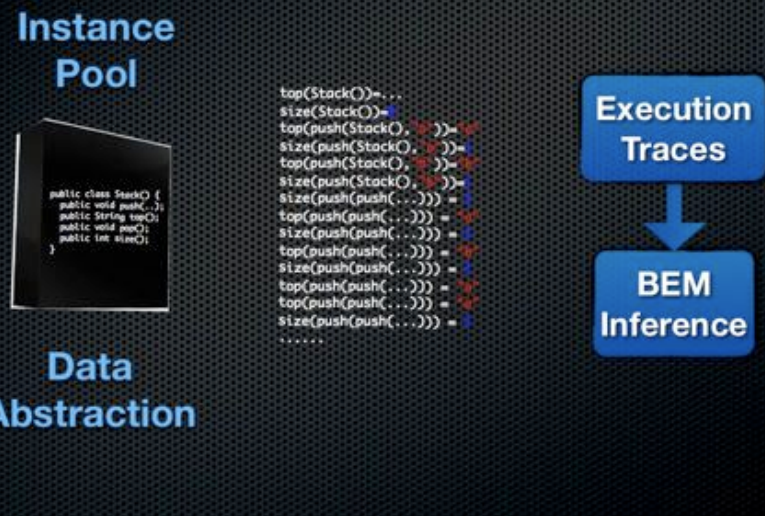
# Intensional Behavior Model




# The Synthesis Process

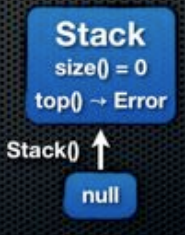



# The Synthesis Process
# 1: BEM Inference



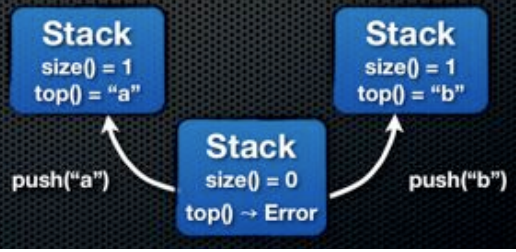

# BEM Inference

```
top(Stack()) throws Error
size(Stack())=0
```



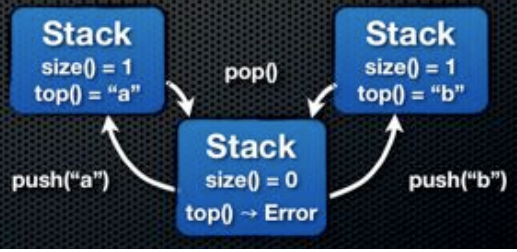
# BEM Inference

```
top(push(Stack(),"a"))="a"
size(push(Stack(),"a"))=1
top(push(Stack(),"b"))="b"
size(push(Stack(),"b"))=1
```



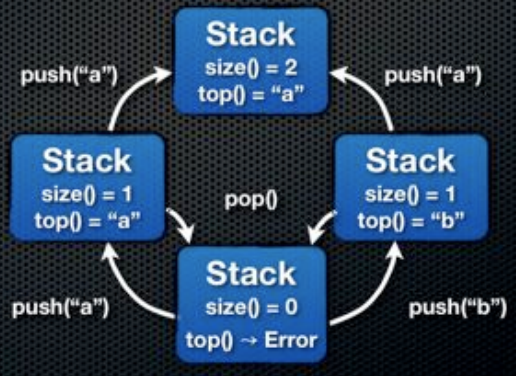

# BEM Inference

```
top(pop(push(Stack(),"a"))) throws Error
size(pop(push(Stack(),"a")))=0
top(pop(push(Stack(),"b"))) throws Error
size(pop(push(Stack(),"b")))=0
```



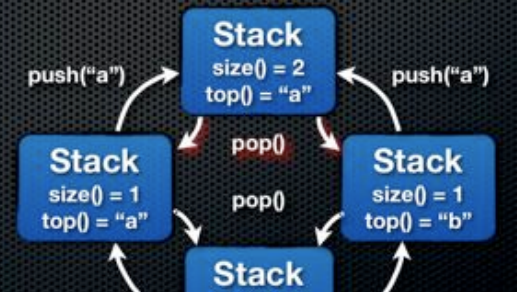

# BEM Inference

```
top(push(push(Stack(),"a"),"a"))="a"
size(push(push(Stack(),"a"),"a"))=2
top(push(push(Stack(),"b"),"a"))="a"
size(push(push(Stack(),"b"),"a"))=2
```




# BEM Inference

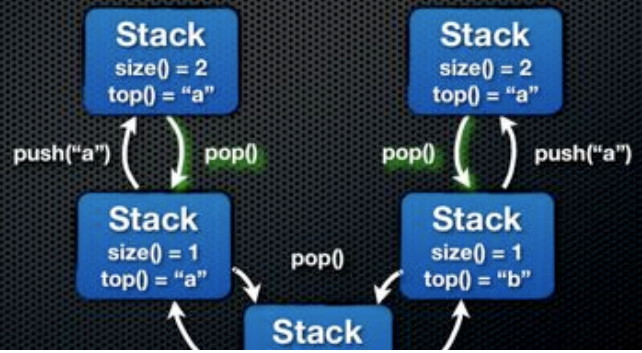
```
top(pop(push(push(Stack(),"a"),"a")))="a"
size(pop(push(push(Stack(),"a"),"a")))=1
top(pop(push(push(Stack(),"b"),"a")))="b"
size(pop(push(push(Stack(),"b"),"a")))=1
```



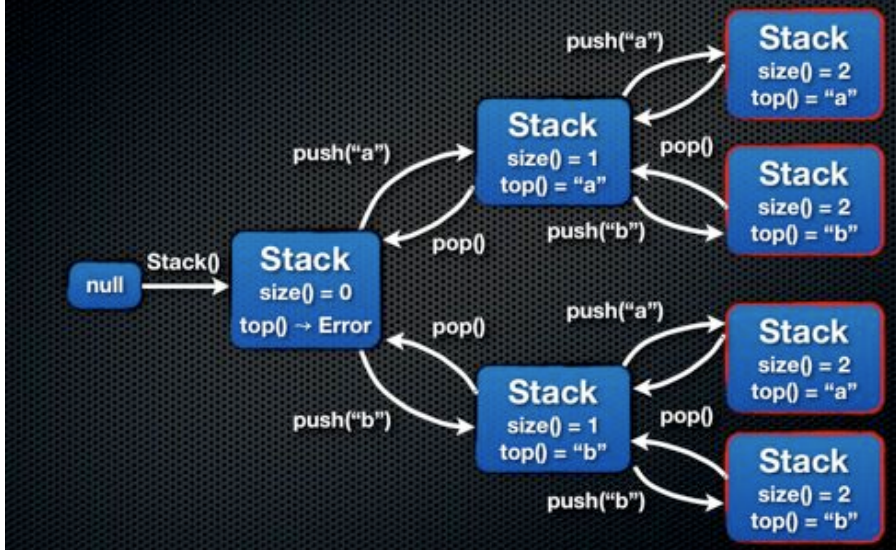
# BEM Inference

```
top(pop(push(push(Stack(),"a"),"a")))="a"
size(pop(push(push(Stack(),"a"),"a")))=1
top(pop(push(push(Stack(),"b"),"a")))="b"
size(pop(push(push(Stack(),"b"),"a")))=1
```

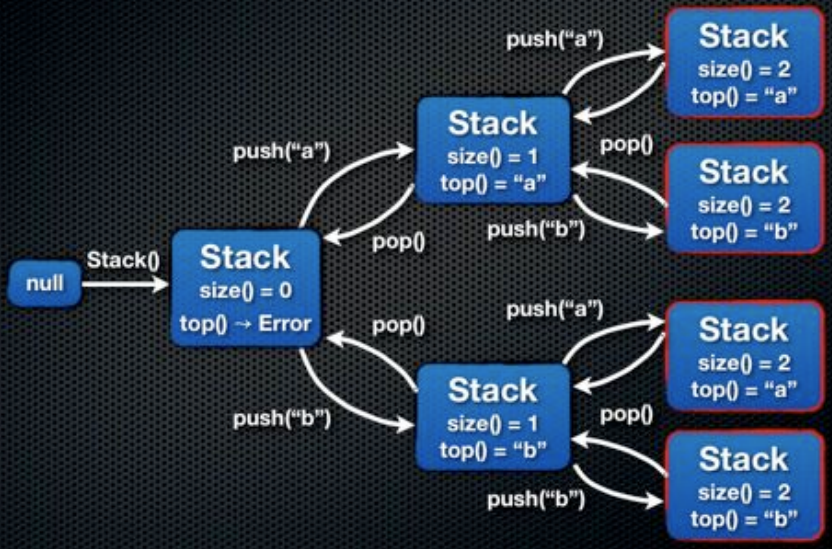
## Discriminating Operation



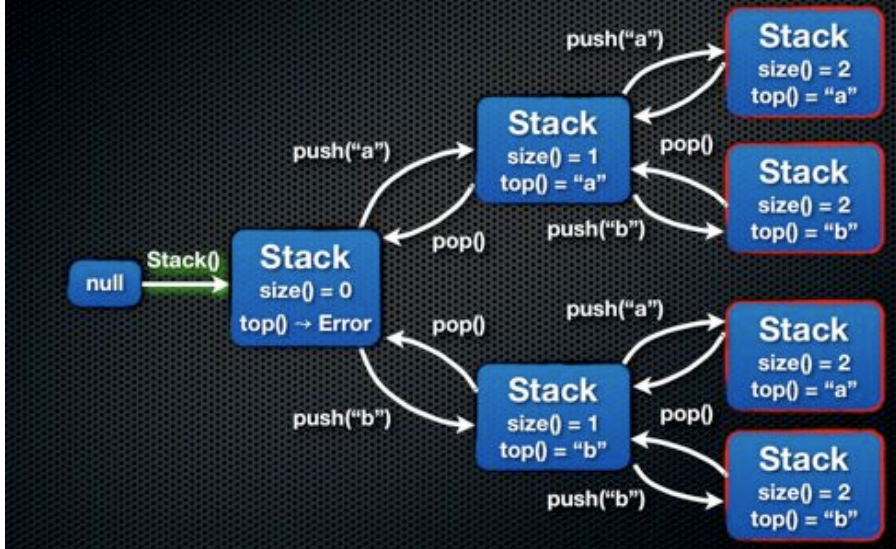

# Inferred BEM



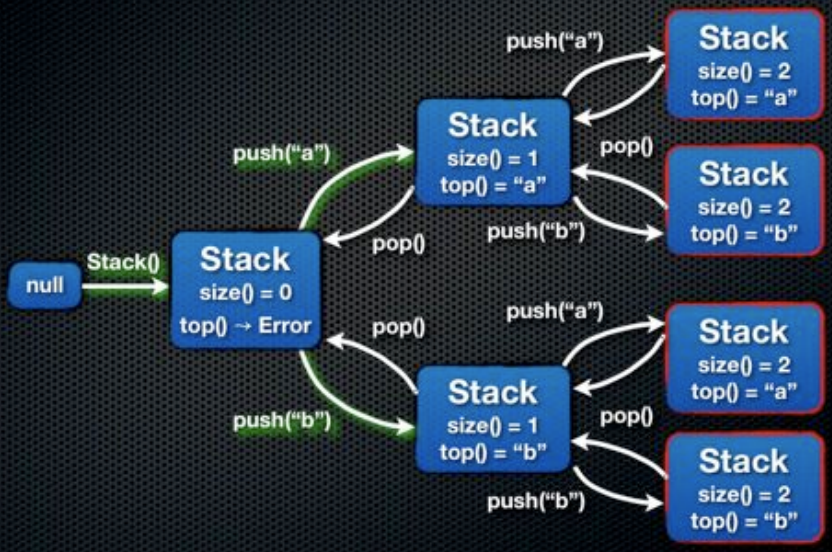

# Canonical Terms




# Canonical Terms

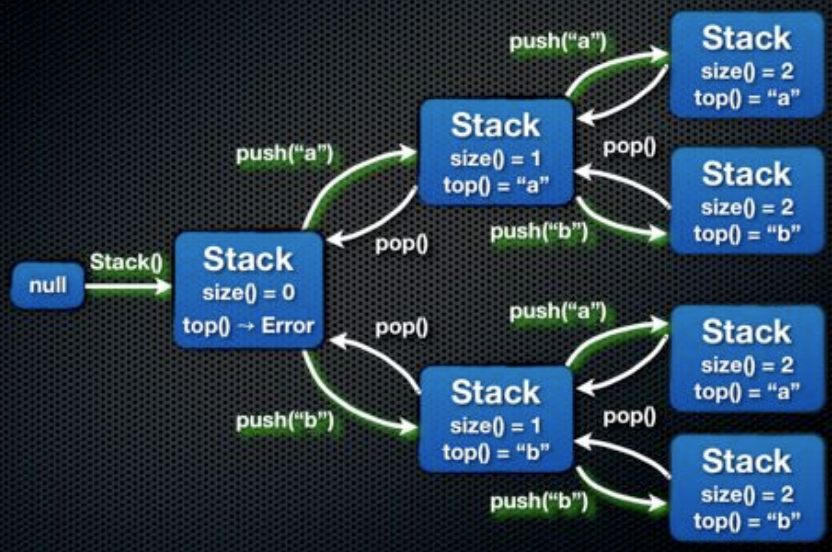


# Canonical Terms

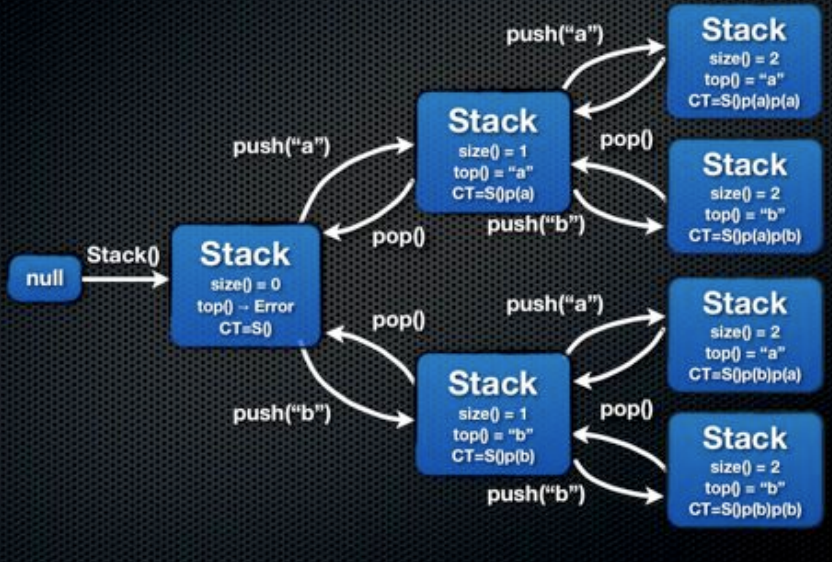




# Canonical Terms


# Inferred BEM




# The Synthesis Process

Instance Pool

Data Abstraction

Execution Traces

BEM Inference

# The Synthesis Process 2: Intensional BM Synthesis

Instance Pool

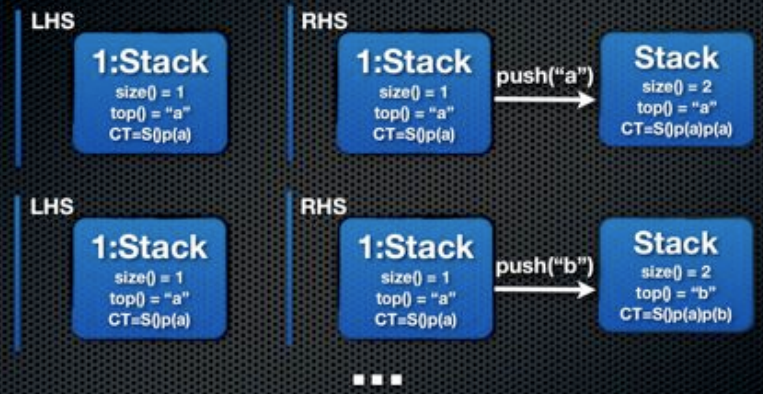
Data Abstraction

Execution Traces

BEM Inference

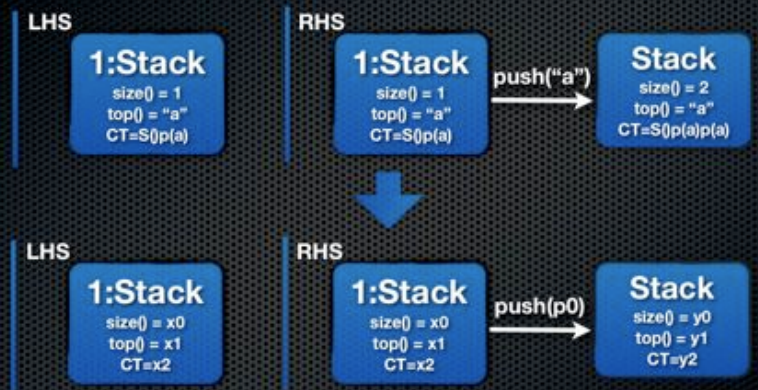
Intensional Behavior Model Synthesis

# Inference Basis



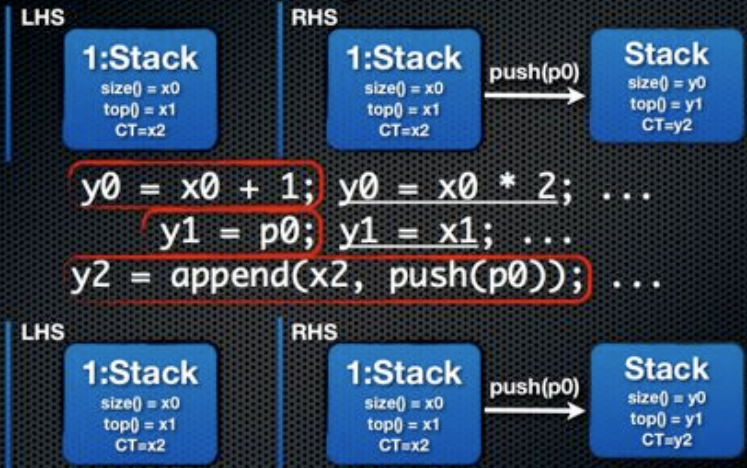

How can I generalize rules?

# Invariant Detection

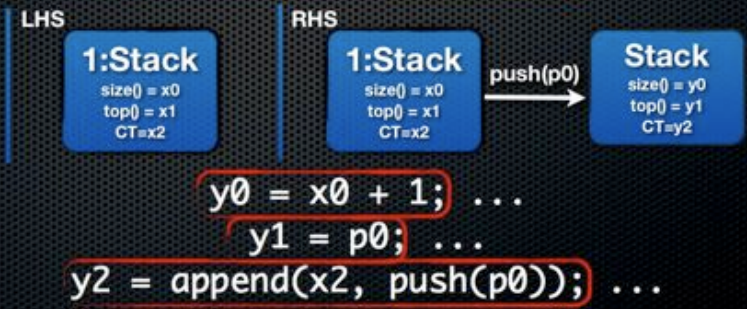



```
y0 = x0 + 1; y0 = x0 * 2; ...
     y1 = p0; y1 = x1; ...
y2 = append(x2, push(p0)); ...
```

# Rule Synthesis

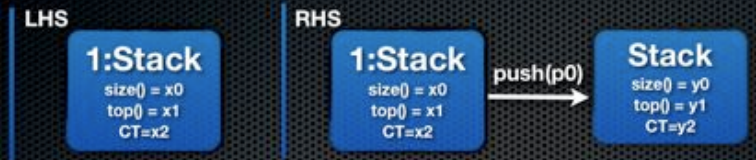



```
y0 = x0 + 1; y0 = x0 * 2; ...
     y1 = p0; y1 = x1; ...
y2 = append(x2, push(p0)); ...
```




```
y0 = x0 + 1; ...
     y1 = p0; ...
y2 = append(x2, push(p0)); ...
```

# Rule Synthesis

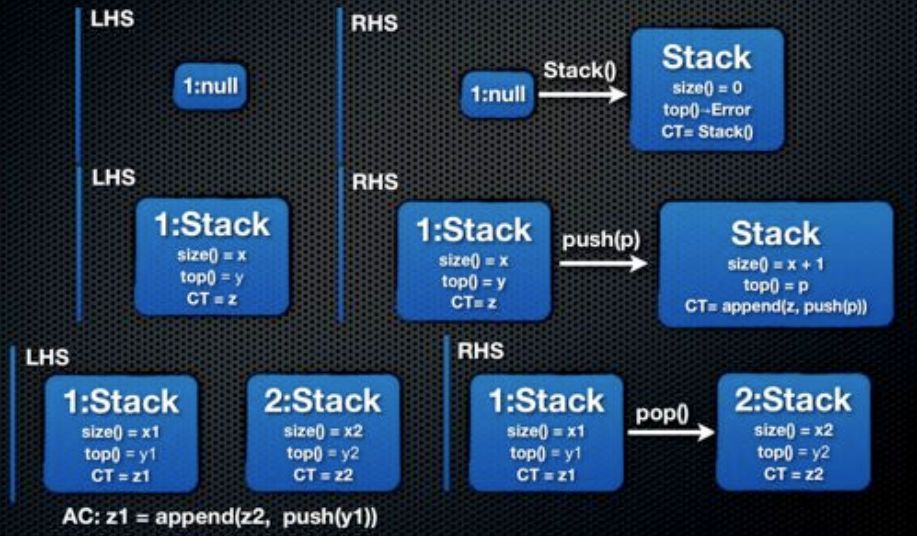



```
y0 = x0 + 1; ...
y1 = p0; ...
y2 = append(x2, push(p0)); ...
```

Invariant Detection also for Preconditions

Ad-hoc Invariants for Canonical Terms

# Intensional Behavior Model




# Related Work

- The pioneer: Daikon [Ernst et al]
- Recent Advances in Invariant Detection: DySy [Csallner et Al]
- Inductive Logic Programming [Sankaranarayanan et Al]
- Temporal API Rules [Acharya et Al]
- Behavior Model Extraction [Dallmeier et Al, Lorenzoli et Al]
- Best Competitive: Heureka [Henkel and Diwan], infers Algebraic Specifications

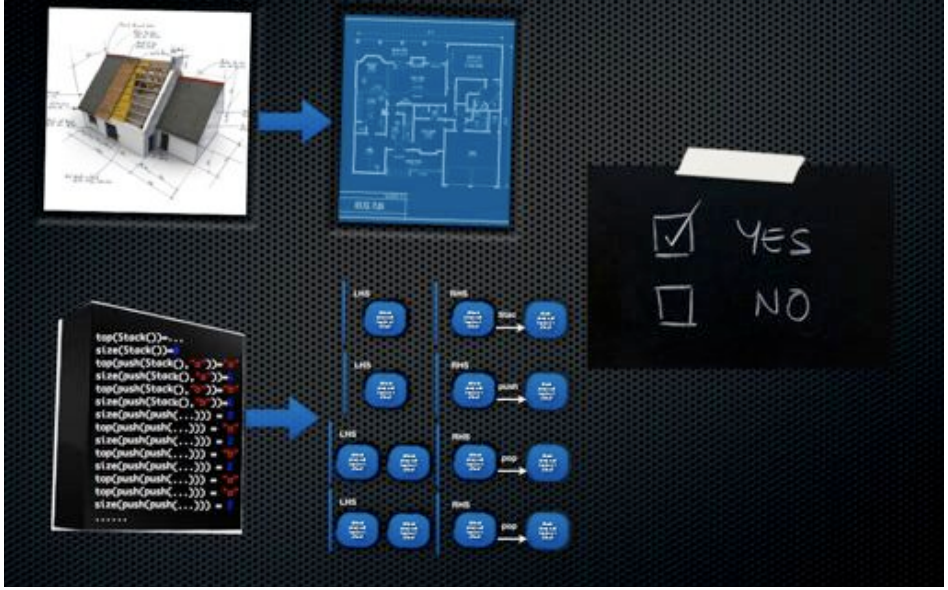
# Evaluation

- Evaluated against Heureka
- Tested with java.util Containers
- Tested with relevant Data Abstractions

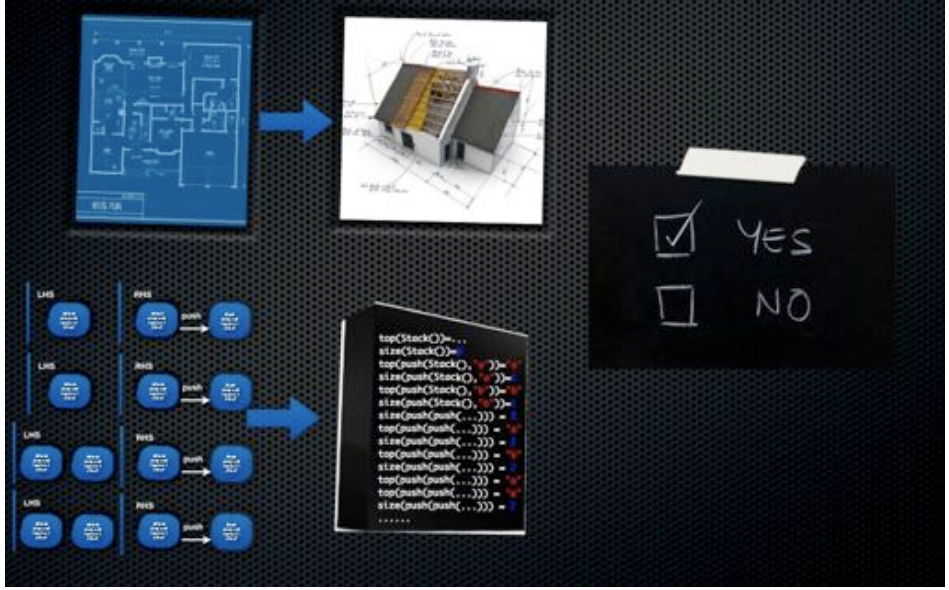
Available at:

http://home.dei.polimi.it/mocci/spy/

# Evaluation




# Evaluation




# Evaluation


# Evaluation

## Evaluation Setting

- Results depend on how many ad-hoc invariants have been added
- We added invariant patterns to cope with specific behaviors of the data abstractions
- Obviously, there exist more complex data abstractions whose spec is hard to infer
- Results have no absolute value, they show a relative improvement
  - i.e., who performs better with those Data Abstractions within the above limits!

## Conclusions

- A method for Specification Recovery through Black-Box Dynamic Analysis
- Works for Containers and Data Abstractions
- Retrieves Precise Specifications for relevant and complex data abstractions
- Several Future Work for Improvement: Nondeterministic Components/Modeling, More General Stateful Components

## Thank you :)