liquid information flow control

Nadia Polikarpova

WG2.8 2021

join work with Deian Stefan, Shachar Itzhaky, Jean Yang, Travis Hance, Armando Solar-Lezama

UC San Diego
COMPUTER SCIENCE & ENGINEERING

The New York Times

Facebook Security Breach Exposes Accounts of 50 Million Users

One of the challenges for Facebook's chief executive Mark Zuckerberg is convincing users that the company handles their data responsibly. Josh Edelson/Agence France-Presse — Getty Images

By Mike Isaac and Sheera Frenkel

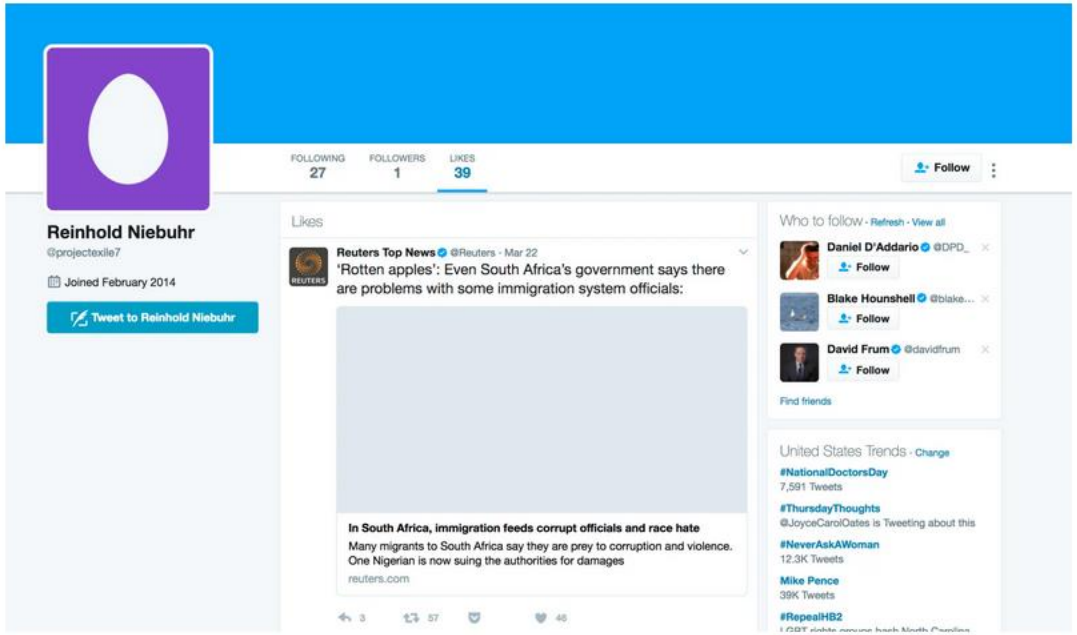
Sept. 28, 2018

SAN FRANCISCO — Facebook, already facing scrutiny over how it handles the private information of its users, said on Friday that an attack on its computer network had exposed the personal information of nearly 50 million users.

PRIVACY AND SECURITY

This Is Almost Certainly James Comey's Twitter Account

Ashley Feinberg
3/30/17 3:29PM • Filed to: JAMES COMEY



Digital security and its discontents—from Hillary Clinton's emails to ransomware to Tor hacks—is in many ways one of the chief concerns of the contemporary FBI. So it makes sense that the bureau's director, James Comey, would dip his toe into the digital torrent with a Twitter account. It also makes sense, given Comey's high profile, that he would want that Twitter account to

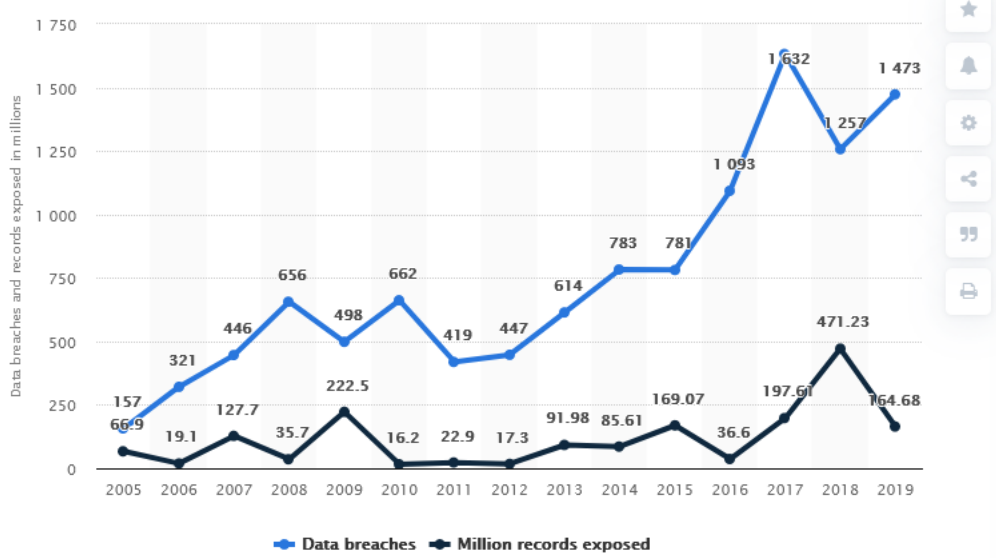
Recent Video

Pink Snow in the Italian Alps Is a Cute Sign of Environmental Catastrophe

Yessenia Funes | 23

Annual number of data breaches and exposed records in the United States from 2005 to 2019

(in millions)



Additional Information | Show source

DOWNLOAD

PDF | XLS | PNG | PPT

Source
→ Show sources information
→ Show publisher information

Release date
January 2020

Region
United States

Survey time period
2005 to 2019

Special properties
sensitive records exposed; excluding non-sensitive records exposed

Cyber crime: number of breaches and records exposed 2005-2019

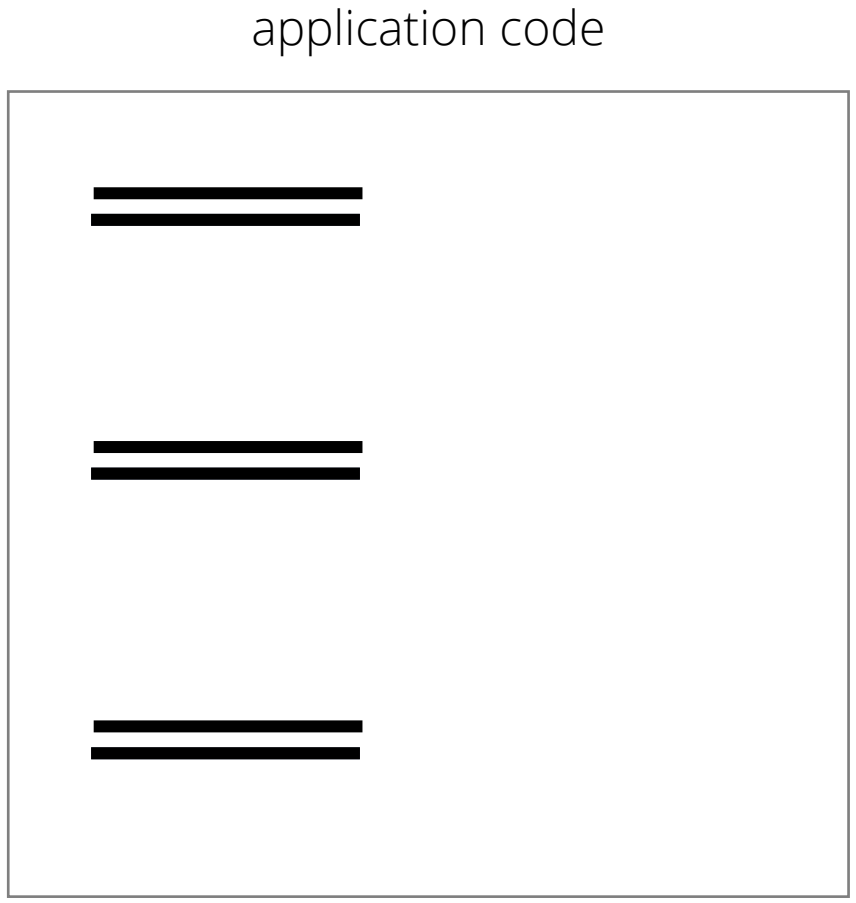
Published by J. Clement, Mar 10, 2020

In 2019, the number of data breaches in the United States amounted to 1,473 with over 164.68 million sensitive records exposed.

Data breaches and exposed records – additional information

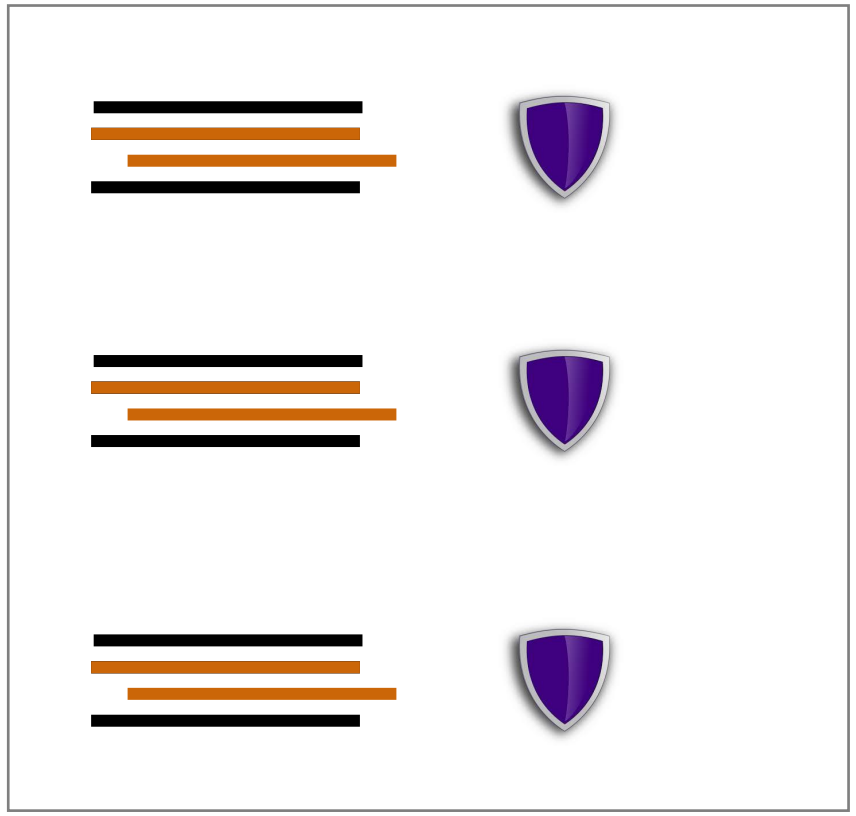
Data breaches have gained attention with the increasing use of digital files and companies and users large reliance on digital data. Even though data breaches happened before digitalization of information – for instance, looking at one's hard copy of medical files without authorization could be

traditional policy enforcement



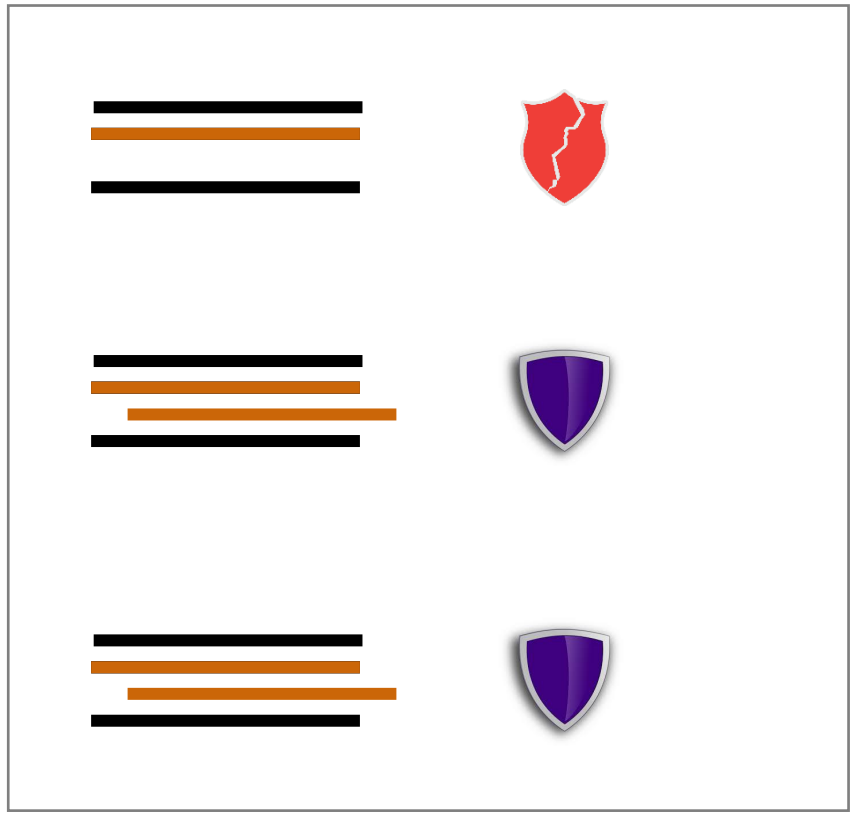
traditional policy enforcement

application code



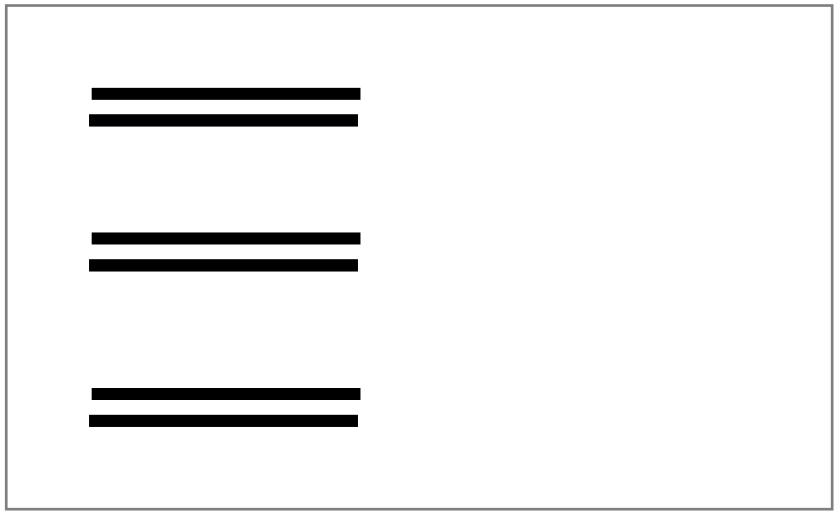
traditional policy enforcement

application code



IFC* frameworks

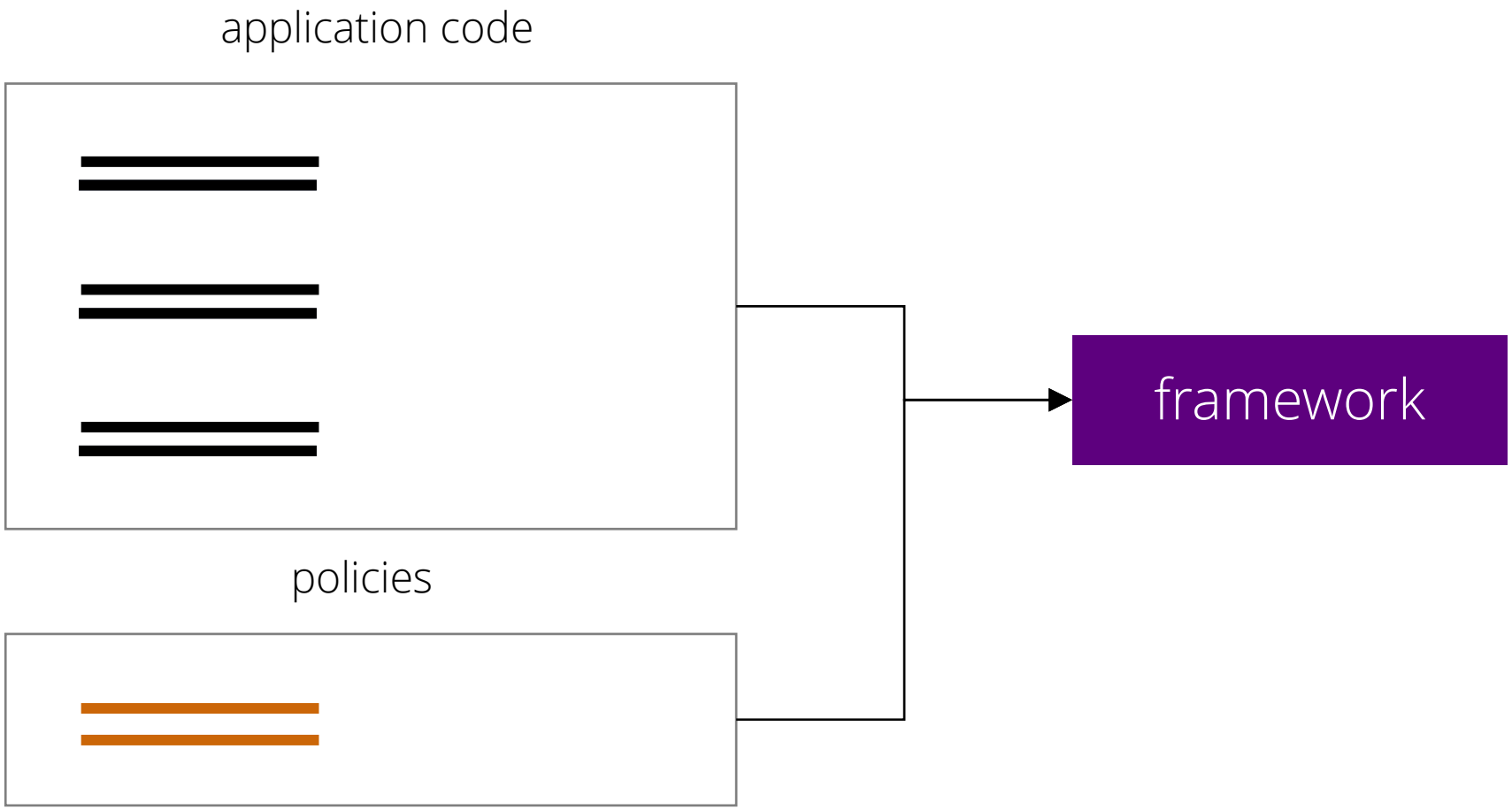
application code



policies

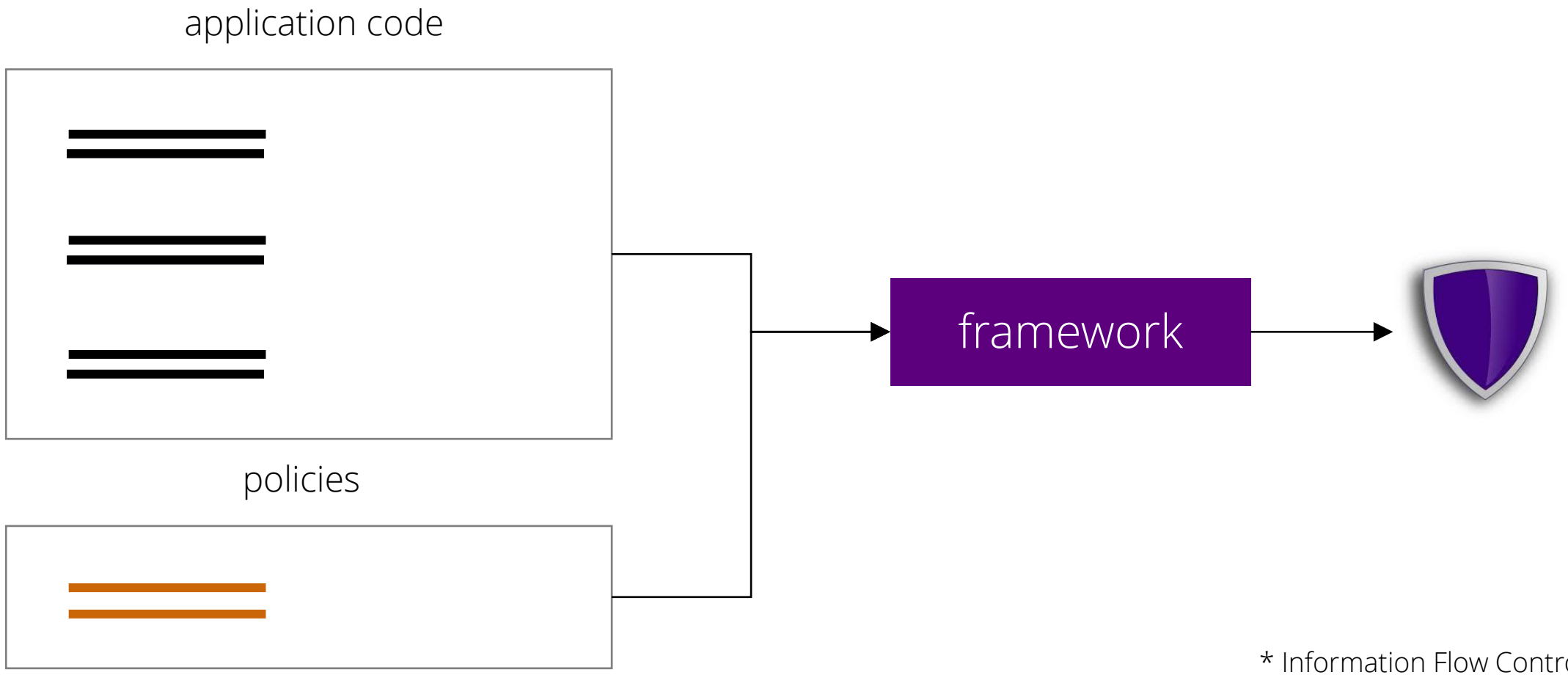
* Information Flow Control

IFC* frameworks



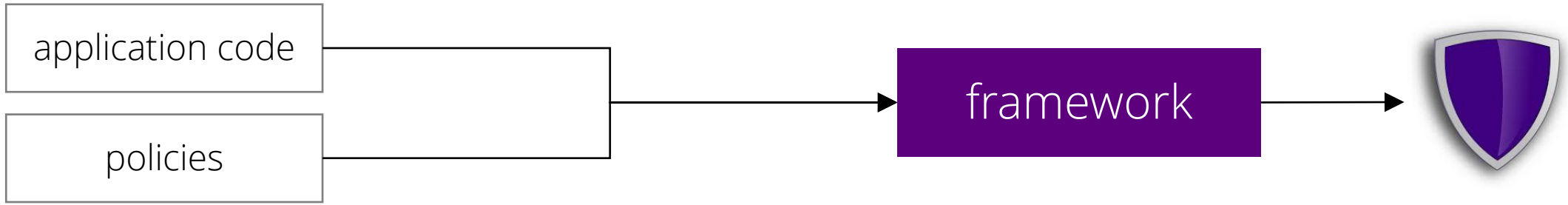
* Information Flow Control

IFC* frameworks



* Information Flow Control

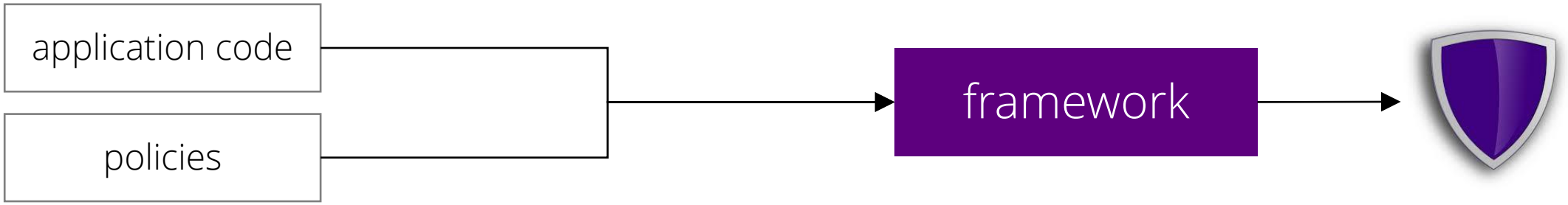
existing IFC frameworks



existing IFC frameworks

...
Fine [Swamy et al 2010]
UrFlow [Chlipala 2010]
HLIO [Buiras et al 2015]

static

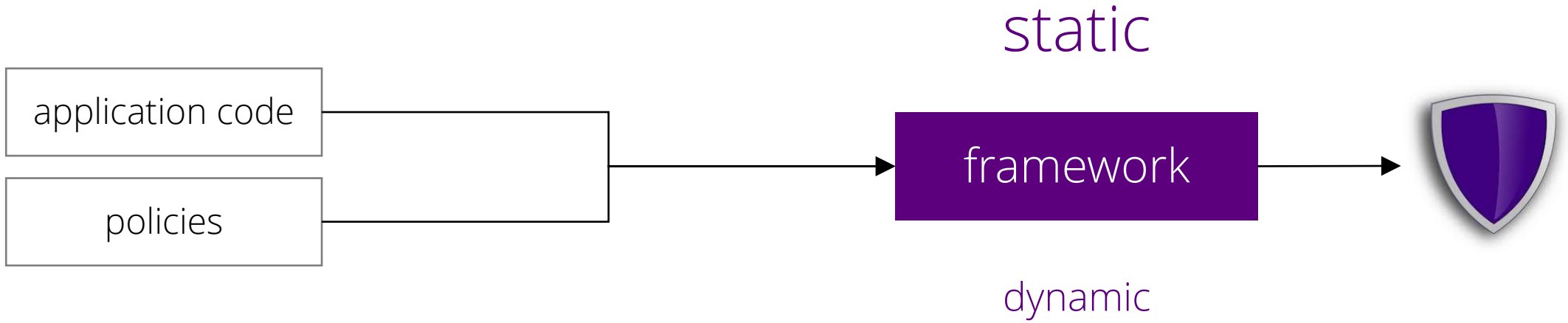


dynamic

LIO [Stefan et al 2011]
Jeeves [Yang et al 2012]
...

existing IFC frameworks

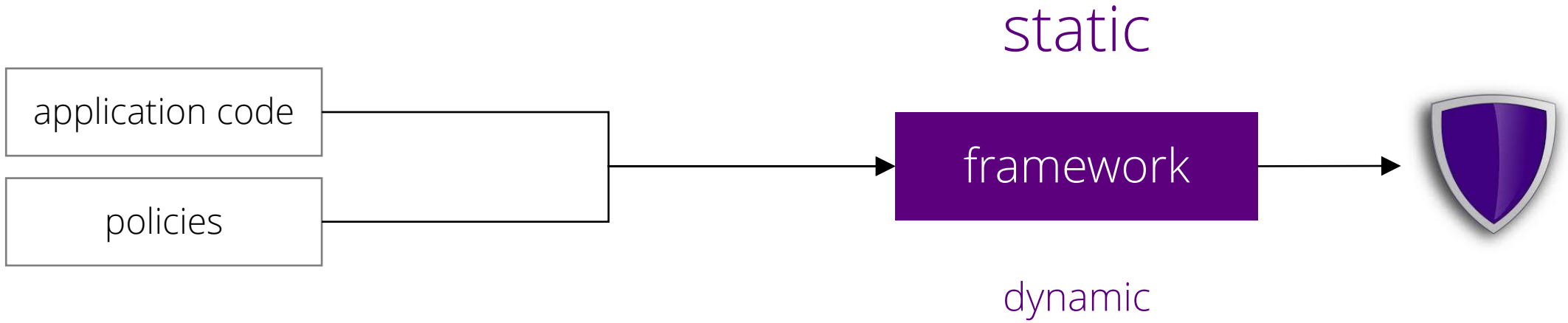
+ predictable behavior
+ no run-time overhead



- crashes / unpredictable behavior
- run-time overhead

existing IFC frameworks

+ predictable behavior
+ no run-time overhead
- rich policies require proof hints



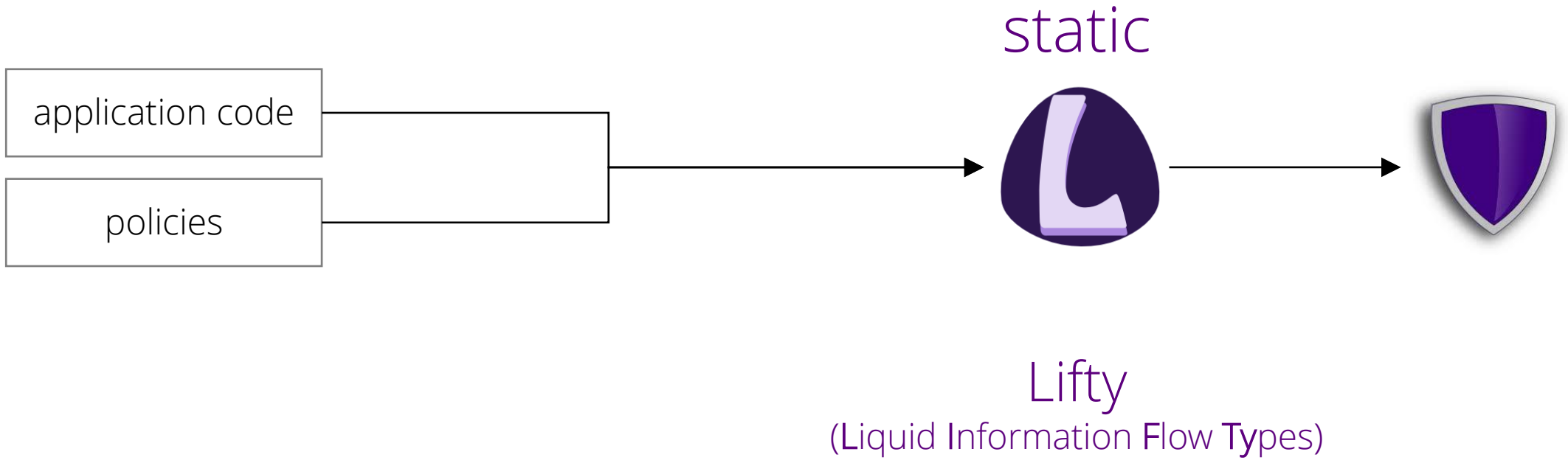
- crashes / unpredictable behavior
- run-time overhead
+ support for rich policies

our solution: lifty

+ predictable behavior
+ no run-time overhead

our solution: lifty

+ predictable behavior
+ no run-time overhead

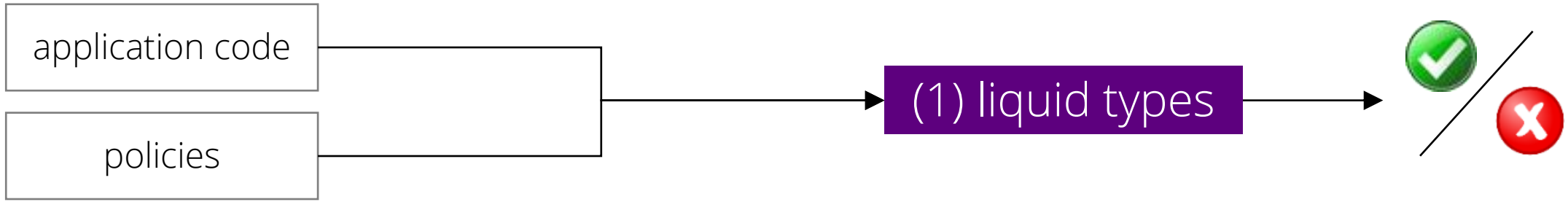


our solution: lifty

+ predictable behavior
+ no run-time overhead
+ automatic checking of rich policies

our solution: lifty

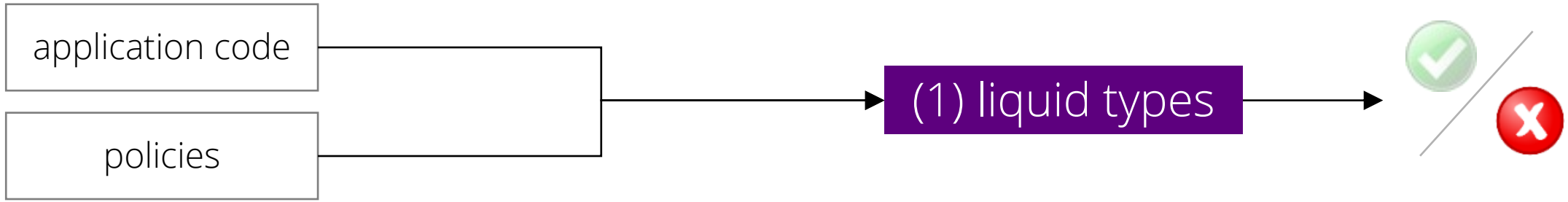
+ predictable behavior
+ no run-time overhead
+ automatic checking of rich policies



Lifty
(Liquid Information Flow Types)

our solution: lifty

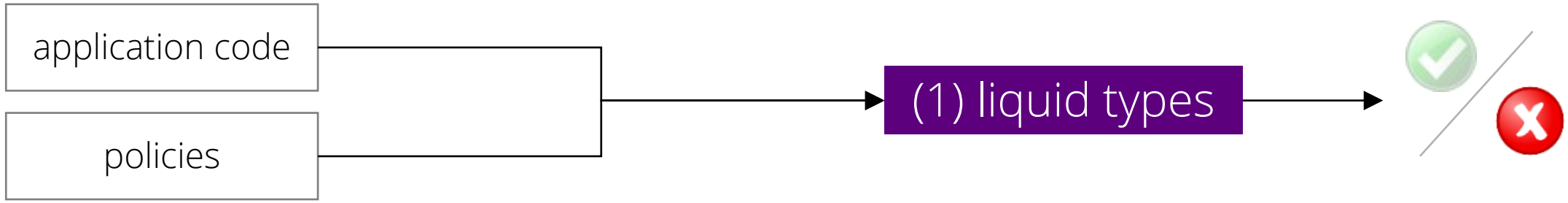
+ predictable behavior
+ no run-time overhead
+ automatic checking of rich policies



Lifty
(Liquid Information Flow Types)

our solution: lifty

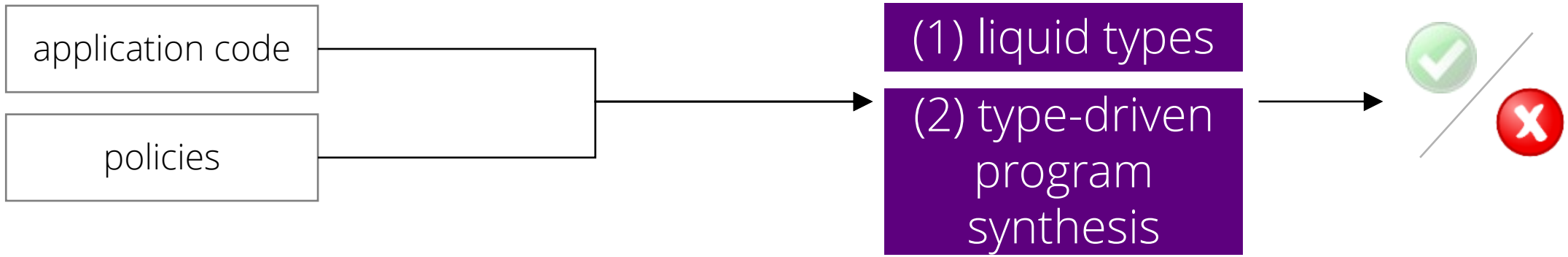
+ predictable behavior
+ no run-time overhead
+ automatic checking of rich policies
+ leak repair



Lifty
(Liquid Information Flow Types)

our solution: lifty

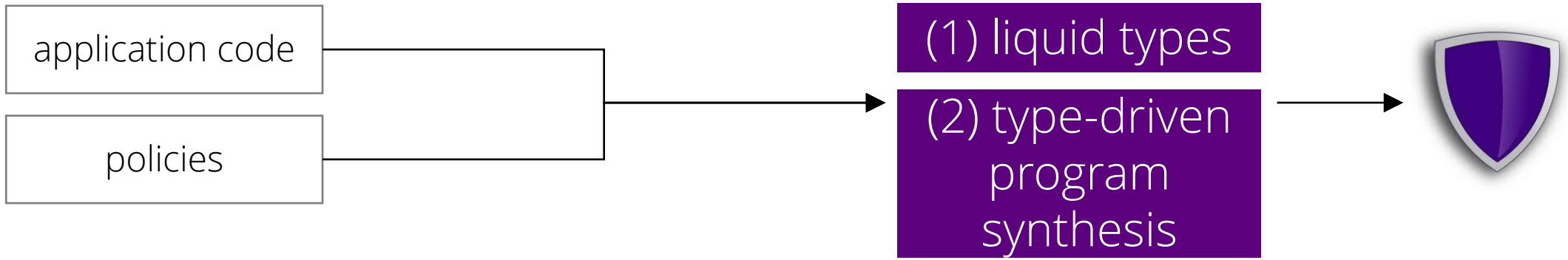
+ predictable behavior
+ no run-time overhead
+ automatic checking of rich policies
+ leak repair



Lifty
(Liquid Information Flow Types)

our solution: lifty

+ predictable behavior
+ no run-time overhead
+ automatic checking of rich policies
+ leak repair



Lifty
(Liquid Information Flow Types)

1. IFC with liquid types

1. IFC with liquid types
2. programming in Lifty

1. IFC with liquid types
2. programming in Lifty
3. leak repair

1. IFC with liquid types
2. programming in Lifty
3. leak repair

liquid types*

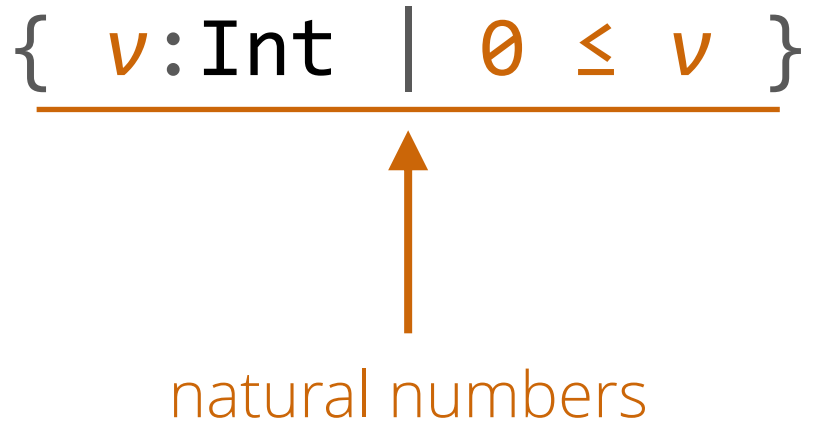
`Int`

* [Rondon, Kawaguchi, Jhala. 2008]

liquid types*

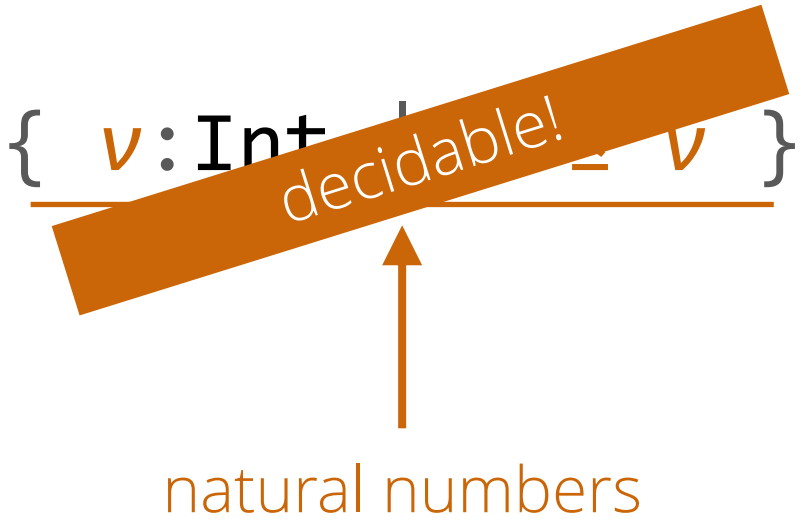
* [Rondon, Kawaguchi, Jhala. 2008]

liquid types*



* [Rondon, Kawaguchi, Jhala. 2008]

liquid types*



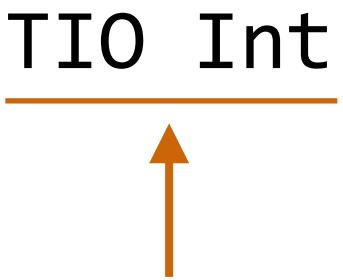
* [Rondon, Kawaguchi, Jhala. 2008]

security monads*

```
TIO Int
```

* [Russo, Claessen, Hughes, 2008], [Stefan et al 2011], ...

security monads*



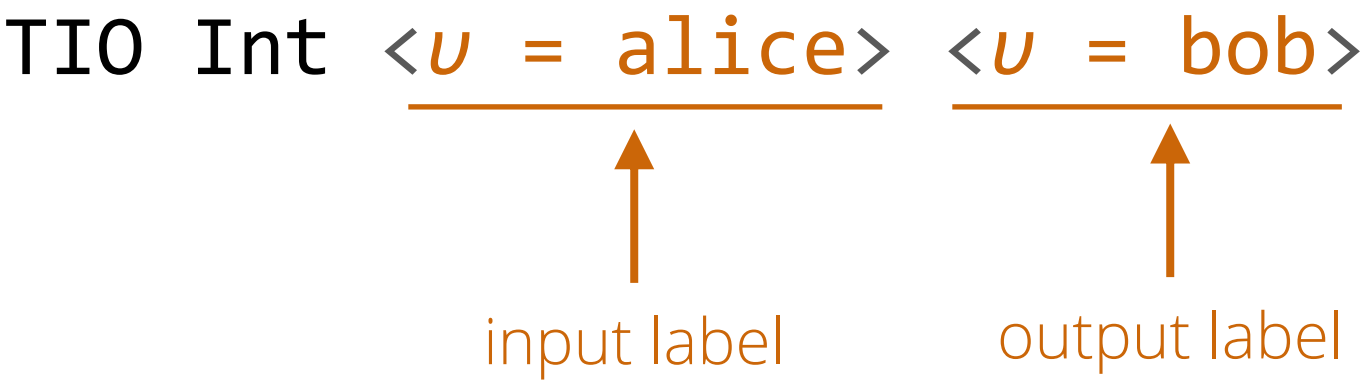
sensitive computation that returns int

* [Russo, Claessen, Hughes, 2008], [Stefan et al 2011], ...

TIO = security monads + liquid types

```
TIO Int <υ = alice> <υ = bob>
```

TIO = security monads + liquid types



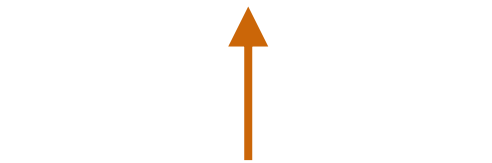
TIO = security monads + liquid types

```
TIO Int <υ = alice> <υ = bob>
```

may read data restricted to alice

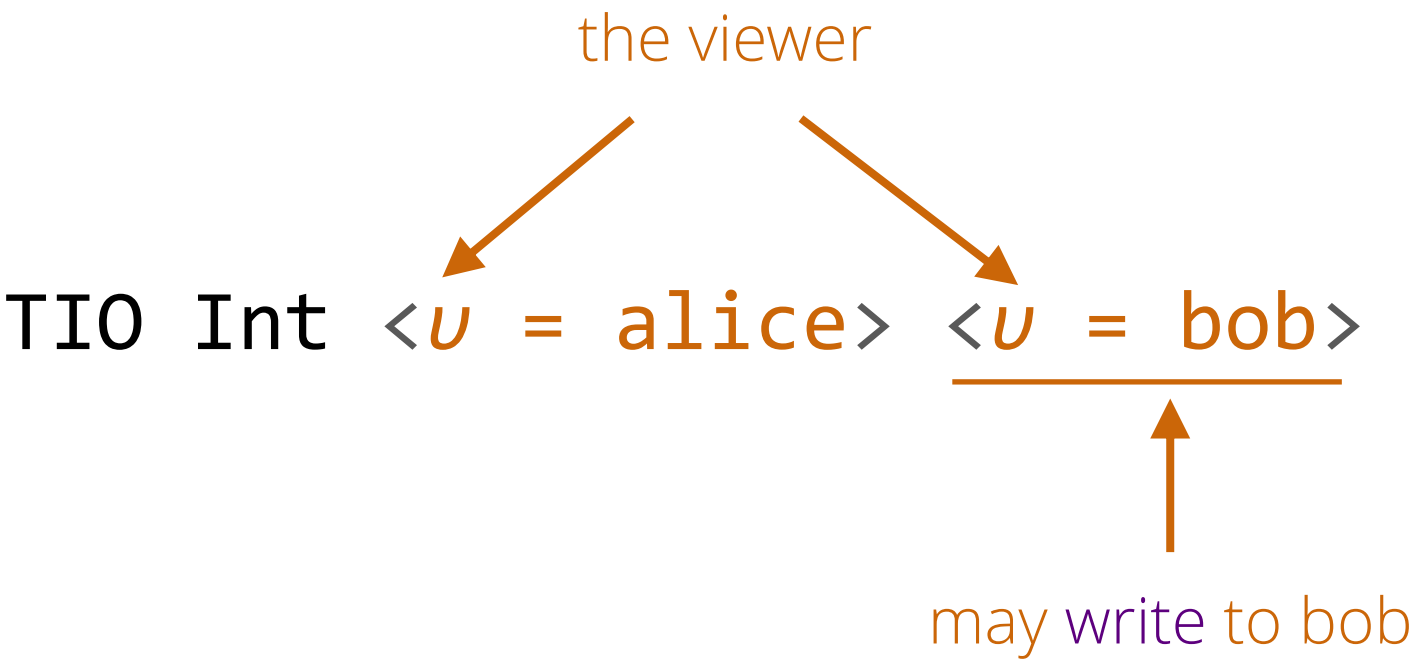
TIO = security monads + liquid types

```
TIO Int <υ = alice> <υ = bob>
```



may write to bob

TIO = security monads + liquid types



1. **IFC with liquid types**
   - the TIO monad
   - building TIO computations
2. **programming in Lifty**
3. **leak repair**

atomic TIO actions

model sources and sinks of sensitive data, e.g.:

atomic TIO actions

model sources and sinks of sensitive data, e.g.:

```
getSharedKey :: TIO String <υ ∈ [alice, bob]> <false>
```

atomic TIO actions

model sources and sinks of sensitive data, e.g.:

```
getSharedKey :: TIO String <υ ∈ [alice, bob]> <false>
```

atomic TIO actions

model sources and sinks of sensitive data, e.g.:

```
getSharedKey :: TIO String <υ ∈ [alice, bob]> <false>
```

atomic TIO actions

model sources and sinks of sensitive data, e.g.:

```
getSharedKey :: TIO String <υ ∈ [alice, bob]> <false>
```

```
print :: rec:User → String → TIO Unit <true> <υ = rec>
```

atomic TIO actions

model sources and sinks of sensitive data, e.g.:

```
getSharedKey :: TIO String <υ ∈ [alice, bob]> <false>
```

```
print :: rec:User → String → TIO Unit <true> <υ = rec>
```

atomic TIO actions

model sources and sinks of sensitive data, e.g.:

```
getSharedKey :: TIO String <υ ∈ [alice, bob]> <false>
```

```
print :: rec:User → String → TIO Unit <true> <υ = rec>
```

atomic TIO actions

model sources and sinks of sensitive data, e.g.:

```
getSharedKey :: TIO String <υ ∈ [alice, bob]> <false>

print :: rec:User → String → TIO Unit <true> <υ = rec>
```

the TIO API

the TIO API

```
-- Return a pure value
return :: a → TIO a
```

the TIO API

```
-- Return a pure value
return :: a → TIO a <true> <false>
```

the TIO API

```
-- Return a pure value
return :: a → TIO a <true> <false>

-- Sequence two sensitive computations
bind :: TIO a →
        (a → TIO b) →
        TIO b
```

the TIO API

```
-- Return a pure value
return :: a → TIO a <true> <false>

-- Sequence two sensitive computations
bind :: TIO a →                          X
        (a → TIO b) →                    Y
        TIO b
```

the TIO API

```
-- Return a pure value
return :: a → TIO a <true> <false>

-- Sequence two sensitive computations
bind :: TIO a <i> <o> →                    X
        (a → TIO b <j> <i ∧ p>) →          Y
        TIO b
```

the TIO API

```
-- Return a pure value
return :: a → TIO a <true> <false>

-- Sequence two sensitive computations
bind :: TIO a <i> <o> →                          X
        (a → TIO b <j> <i ∧ p>) →                Y
        TIO b
```

the TIO API

```
-- Return a pure value
return :: a → TIO a <true> <false>

-- Sequence two sensitive computations
bind :: TIO a <i> <p> →
        (a → TIO b <j> <i ∧ p>) →
        TIO b
```

X

Y

the TIO API

```
-- Return a pure value
return :: a → TIO a <true> <false>

-- Sequence two sensitive computations
bind :: TIO a <i> <o> →                    X
        (a → TIO b <j> <i ∧ p>) →          Y
        TIO b
```

the TIO API

```
-- Return a pure value
return :: a → TIO a <true> <false>

-- Sequence two sensitive computations
bind :: TIO a <i> <o> →                          X
        (a → TIO b <j> <i ∧ p>) →                Y
        TIO b <i ∧ j>
```

the TIO API

```
-- Return a pure value
return :: a → TIO a <true> <false>

-- Sequence two sensitive computations
bind :: TIO a <i> <o> →                          X
        (a → TIO b <j> <i ∧ p>) →                Y
        TIO b <i ∧ j> <o ∨ (i ∧ p)>
```

the TIO API

```
-- Return a pure value
return :: a → TIO a <true> <false>

-- Sequence two sensitive computations
bind :: TIO a <i> <o> →
        (a → TIO b <j> <i ∧ p>) →
        TIO b <i ∧ j> <o ∨ (i ∧ p)>

-- Safely downgrade a Boolean
downgrade :: … -- in the paper
```

the TIO API

```
return :: a → TIO a <true> <false>

bind :: TIO a <i> <o> →
        (a → TIO b <j> <i ∧ p>) →
        TIO b <i ∧ j> <o ∨ (i ∧ p)>

downgrade :: …
```

```
liftM :: …          filterM :: …
mapM :: …           sortM :: …
```

the TIO API

```
return :: a → TIO a <true> <false>

bind :: TIO a <i> <o> →
        (a → TIO b <j> <i ∧ p>) →
        TIO b <i ∧ j> <o ∨ (i ∧ p)>

downgrade :: …
```

primitive operations
(trusted)

```
liftM :: …          filterM :: …
mapM :: …           sortM :: …
```

the TIO API

```
return :: a → TIO a <true> <false>

bind :: TIO a <i> <o> →
        (a → TIO b <j> <i ∧ p>) →
        TIO b <i ∧ j> <o ∨ (i ∧ p)>

downgrade :: …
```

primitive operations (trusted)

```
liftM :: …          filterM :: …
mapM :: …           sortM :: …
```

derived operations (verified)

1. IFC with liquid types
2. programming in Lifty
3. leak repair

running example: conference manager

running example: conference manager

#	title*	score	decision
1	Non-standard Paraconsistent Calculus of Coinductive Constructions with Isorecursive μ-types	1.8	accept
2	Naive Gradual Π-Calculus with Strong Normalization	0.8	reject
3	Abstract Impredicative Type Theory with Ownership Types	2.0	accept
4	Higher Type Theory with Primitive Recursion	1.5	accept
5	Liquid Information Flow Control	C	C
6	Standard Higher Calculus of Inductive Constructions with Full Type Inference	0.5	reject

*titles generated by http://type.systems/

running example: conference manager

#	title*	score	decision
1	Non-standard Paraconsistent Calculus of Coinductive Constructions with Isorecursive μ-types	1.8	accept
2	Naive Gradual Π-Calculus with Strong Normalization	0.8	reject
3	Abstract Impredicative Type Theory with C		pt
4	Higher Type	1.5	accept
5	Liquid Information Flow Control	c	c
6	Standard Higher Calculus of Inductive Constructions with Full Type Inference	0.5	reject

policy: if conflicted, can't see score until notification

*titles generated by http://type.systems/

running example: conference manager

#	title*	score	decision
1	Non-standard Paraconsistent Calculus of Coinductive Constructions with Isorecursive μ-types	1.8	accept
2	Naive Gradual Π-Calculus with Strong Normalization	0.8	reject
3	Abstract Impredicative Type Theory with Ownership Types	2.0	accept
4	Higher Type Theory with Primitive Recursion	1.5	accept
5	Liquid Information Flow Control	C	C
6	Standard Higher Calculus of Inductive Constructions with Full Type Inference	0.5	reject

*titles generated by http://type.systems/

running example: conference manager

#	title*	score ▽	decision
3	Abstract Impredicative Type Theory with Ownership Types	2.0	accept
5	Liquid Information Flow Control	c	c
1	Non-standard Paraconsistent Calculus of Coinductive Constructions with Isorecursive μ-types	1.8	accept
4	Higher Type Theory with Primitive Recursion	1.5	accept
2	Naive Gradual Π-Calculus with Strong Normalization	0.8	reject
6	Standard Higher Calculus of Inductive Constructions with Full Type Inference	0.5	reject

*titles generated by http://type.systems/

running example: conference manager

#	title*	score ▽	decision
3	Abstract Impredicative Type Theory with Ownership Types	2.0	accept
5	Liquid Information Flow Control	c	c
1	Non-standard Paraconsistent Calculus of Coinductive Constructions with Isorec		pt
4	Higher Type	1.5	accept
2	Naive Gradual IT-Calculus with Strong Normalization	0.8	reject
6	Standard Higher Calculus of Inductive Constructions with Full Type Inference	0.5	reject

leaks score through order of papers!

*titles generated by http://type.systems/

demo

1. IFC with liquid types
2. programming in Lifty
3. leak repair

example

```
do
  x ← getPaperScore ds pid
  print client (show x)
```

example

how does Lifty repair this leak?

```
do
  x ← getPaperScore ds pid
  print client (show x)
```

example

how does Lifty repair this leak?

```
do
  x ← getPaperScore ds pid
  print client (show x)
```

key insight:
use type errors to localize leaks!

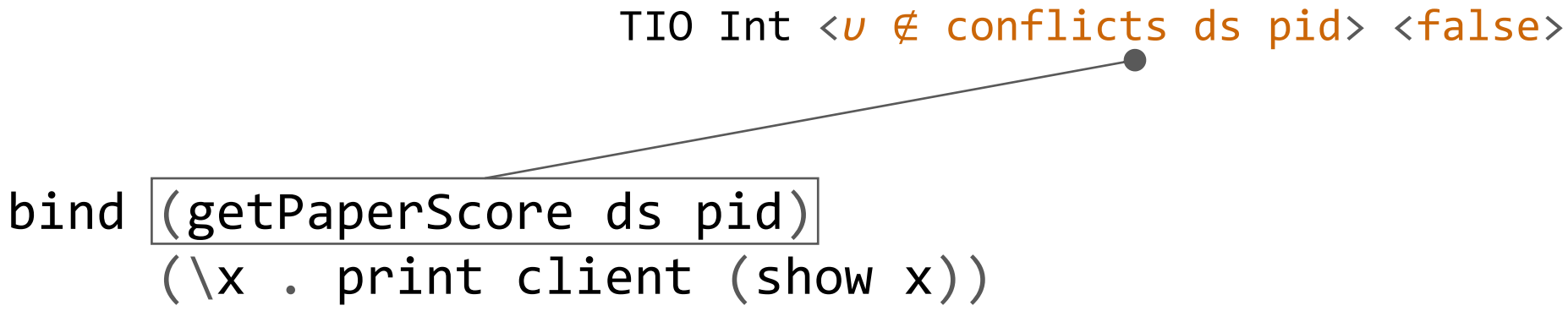
example

```
bind (getPaperScore ds pid)
     (\x . print client (show x))
```

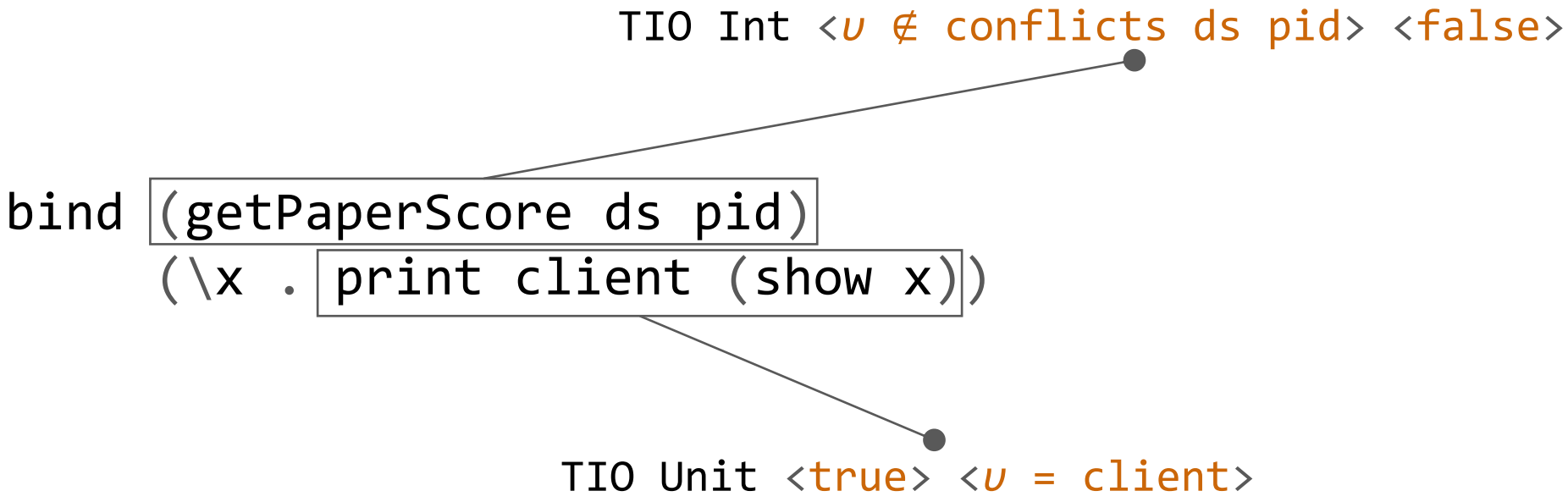
step 1: liquid type checking

```
bind (getPaperScore ds pid)
     (\x . print client (show x))
```

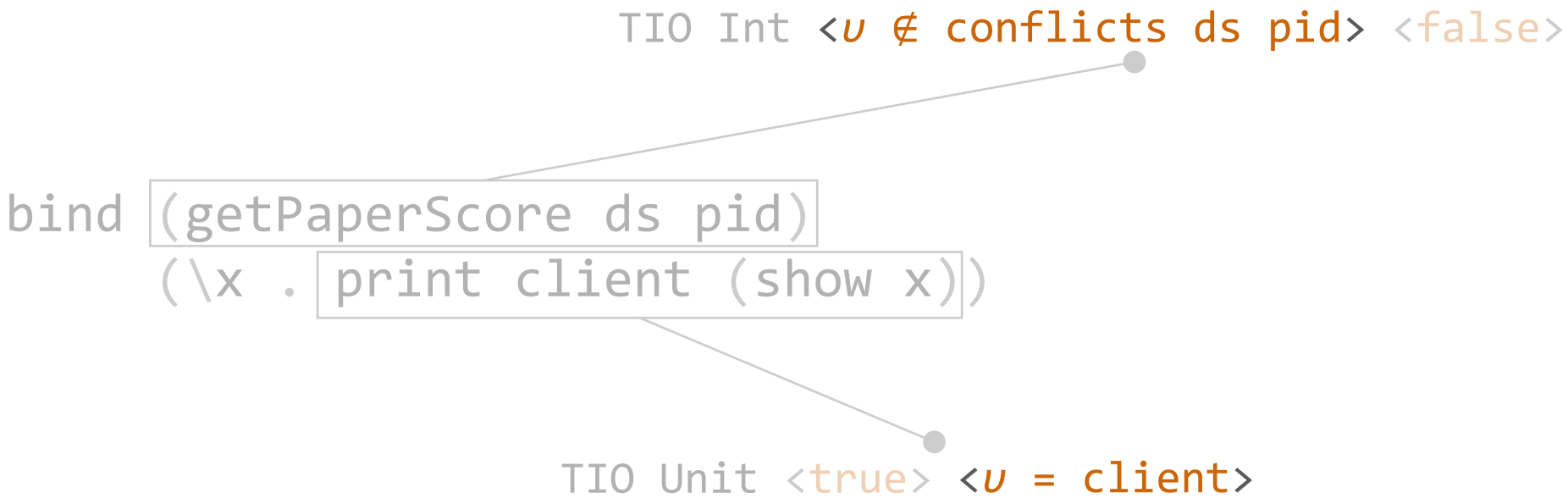
step 1: liquid type checking



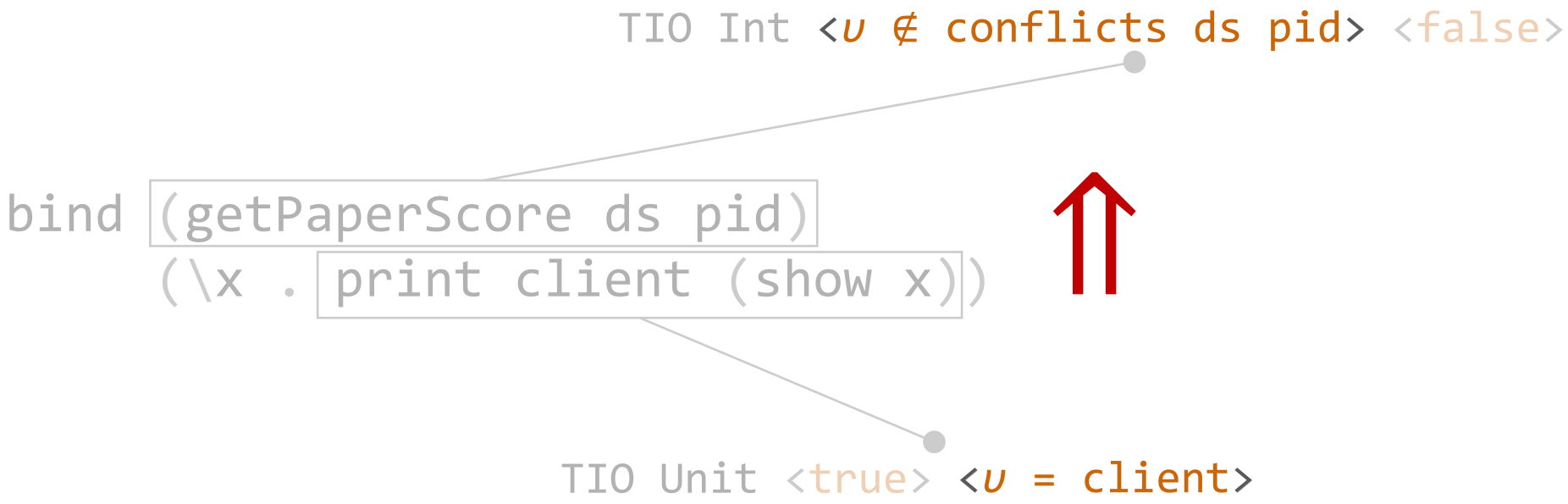
step 1: liquid type checking



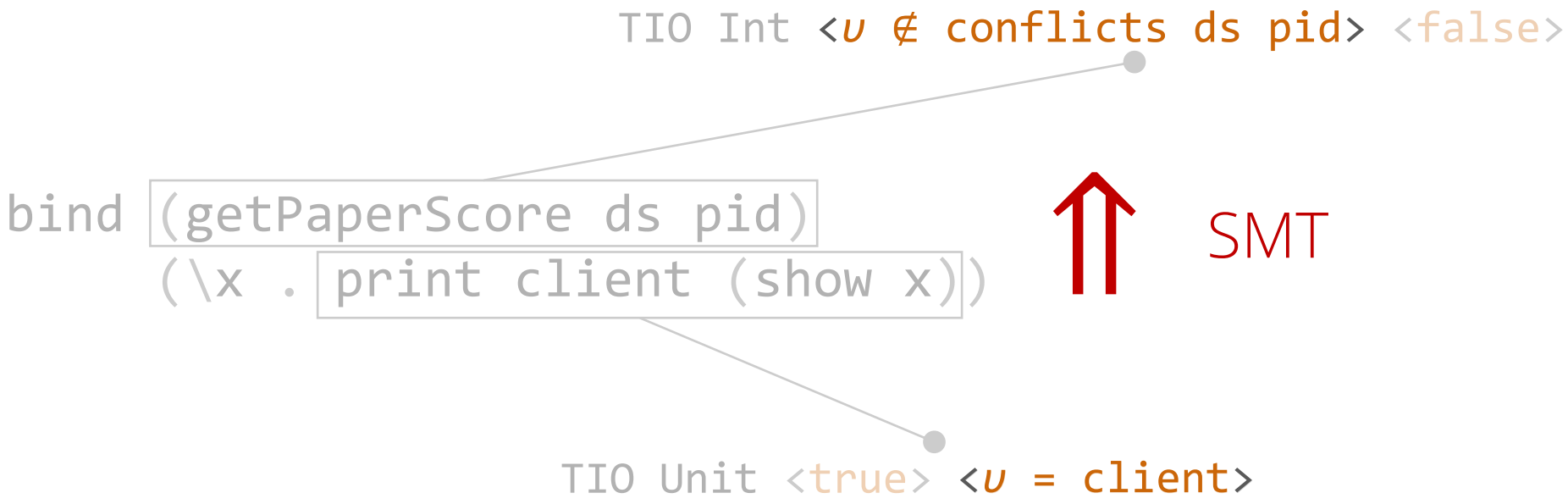
step 1: liquid type checking



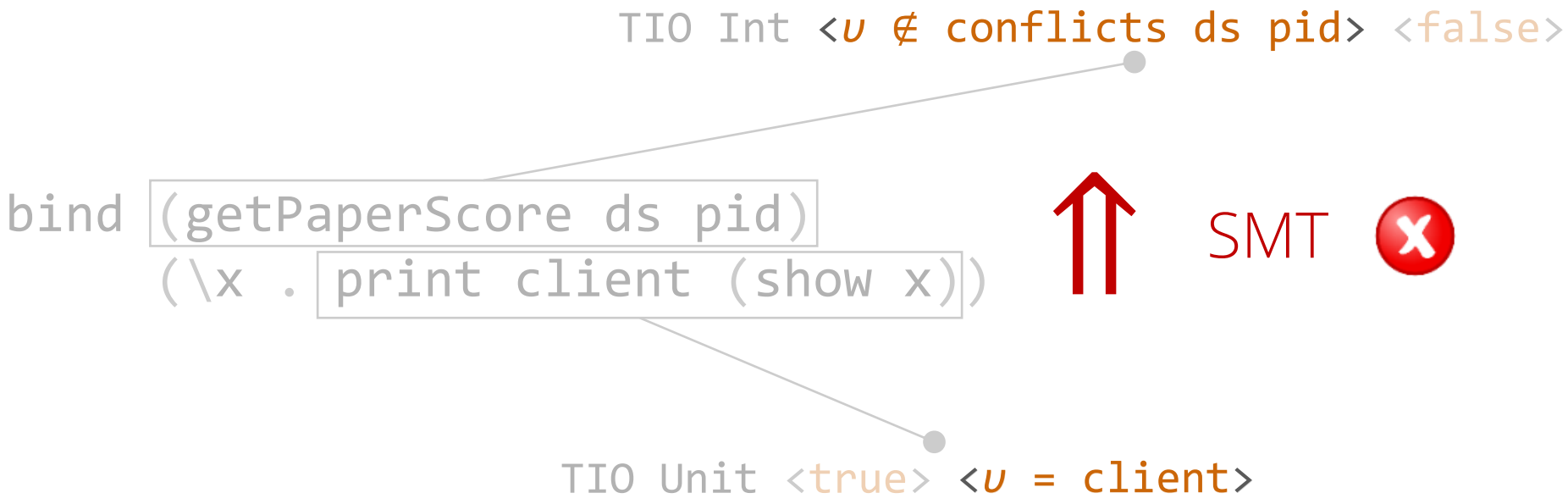
step 1: liquid type checking



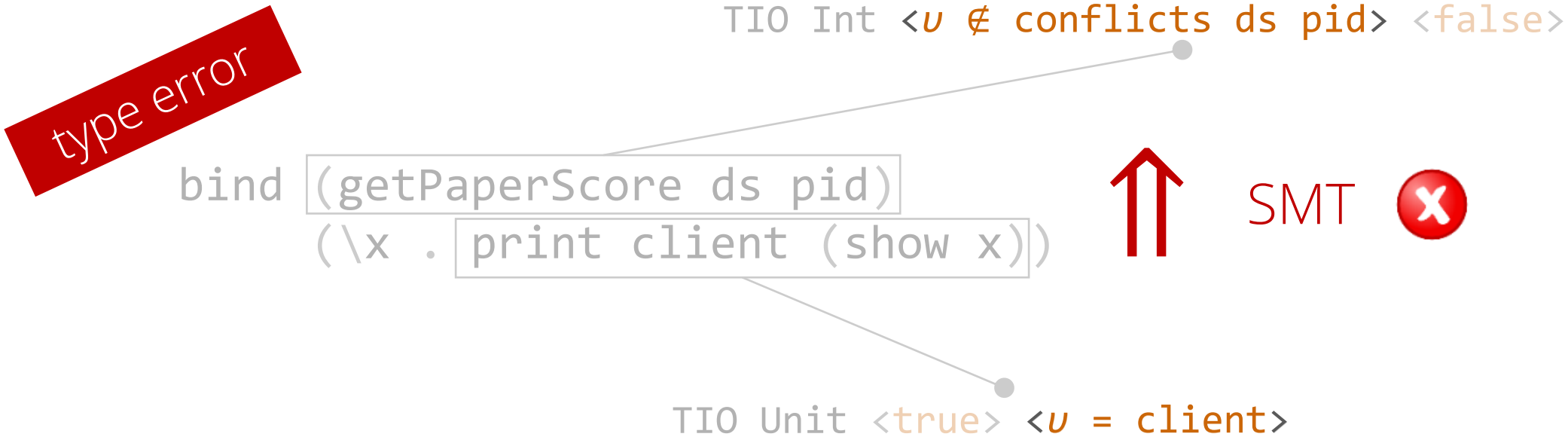
step 1: liquid type checking



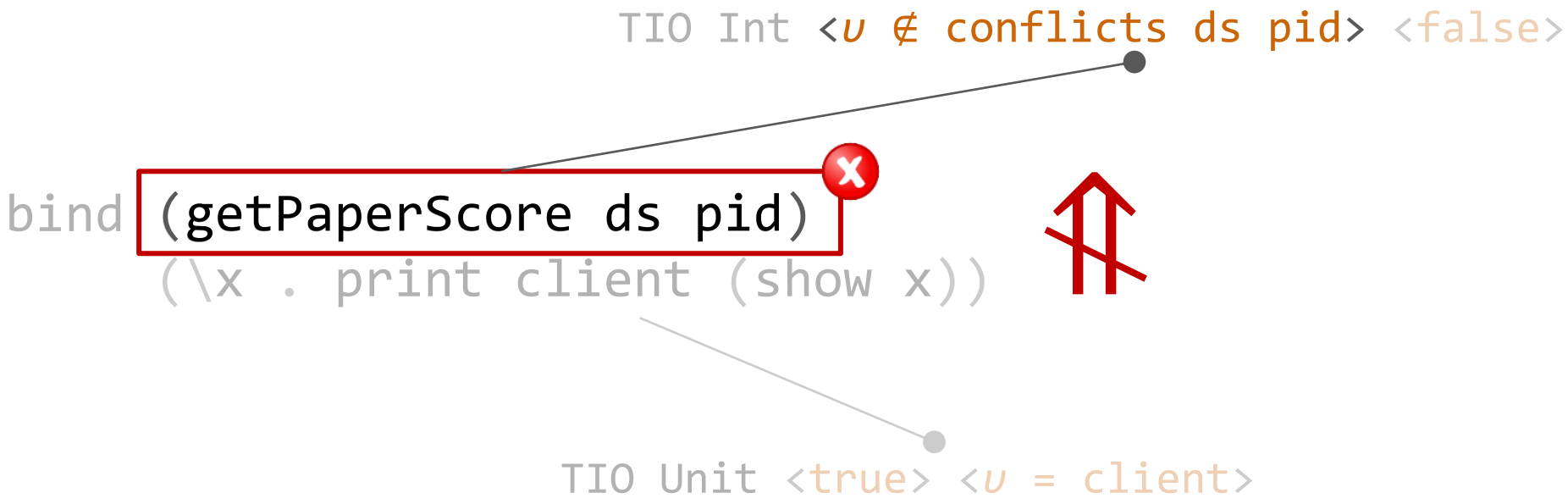
step 1: liquid type checking



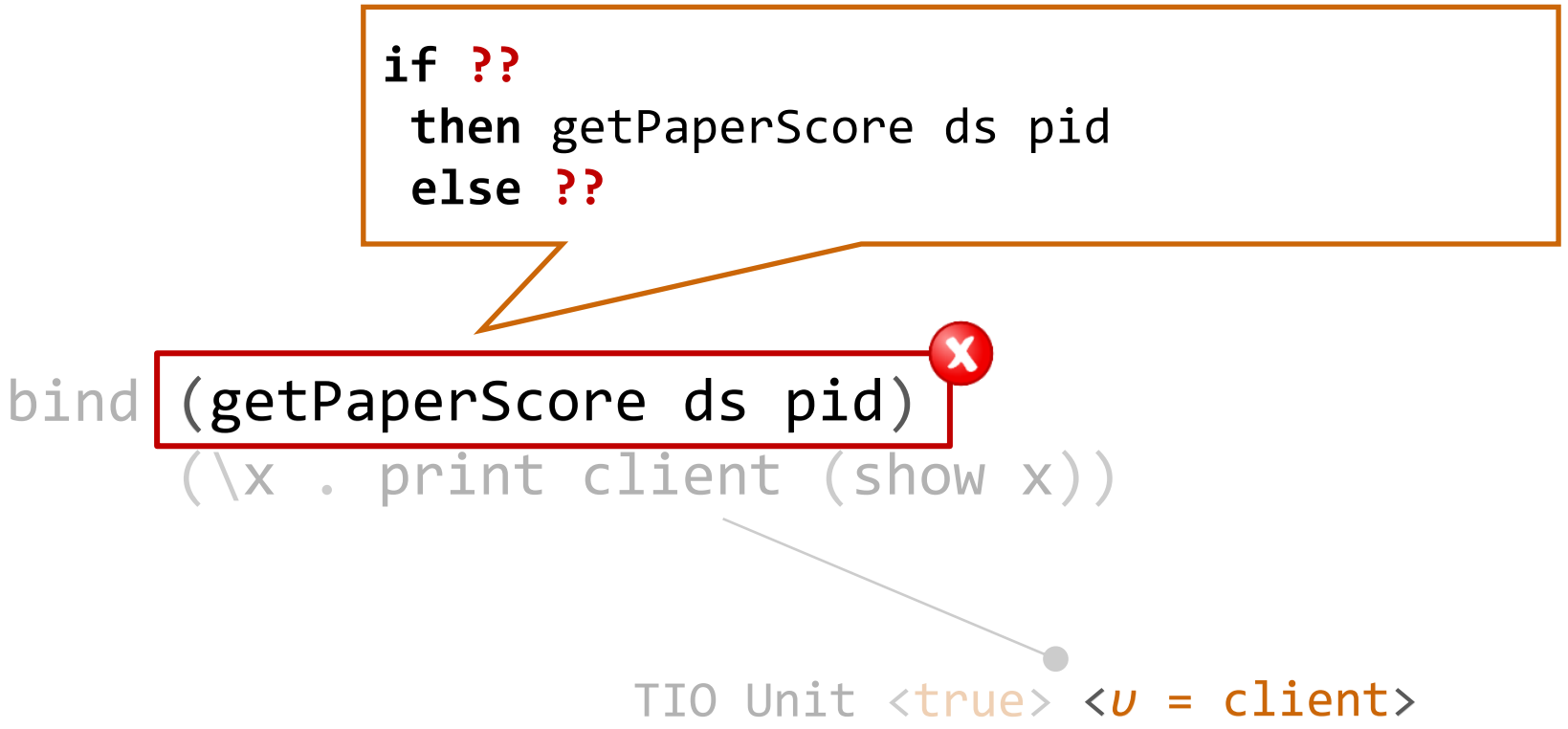
step 1: liquid type checking



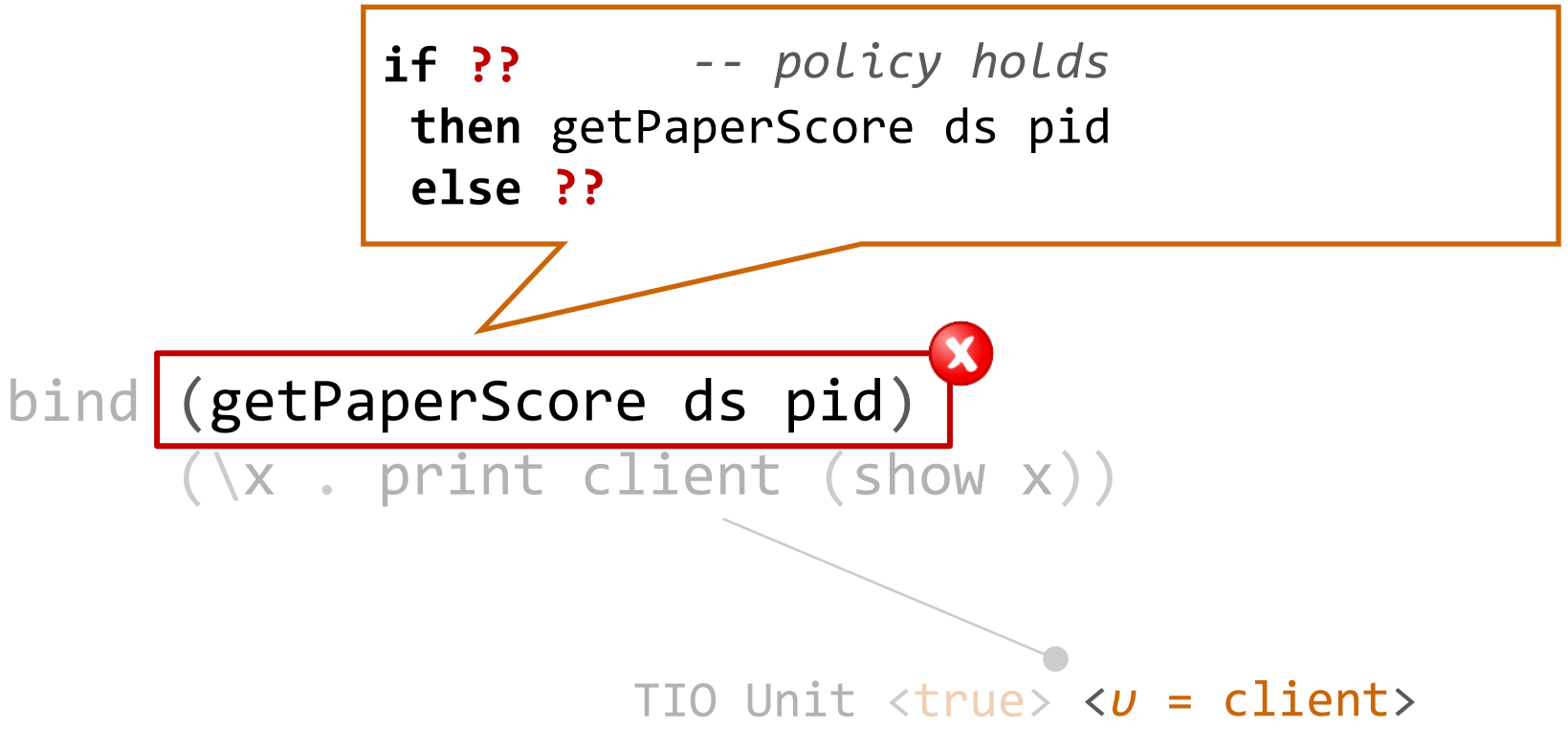
step 2: error localization



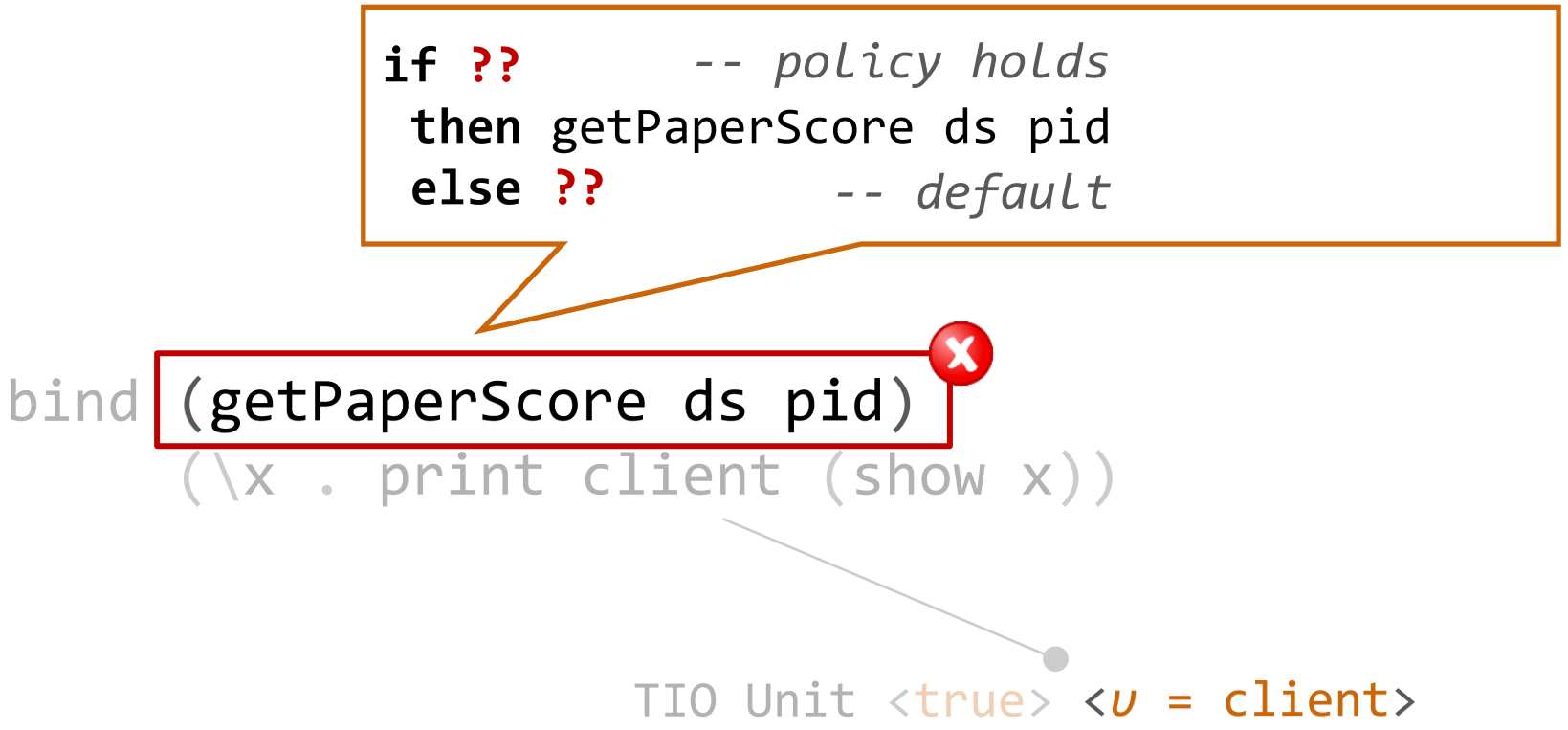
step 3: infer local specification



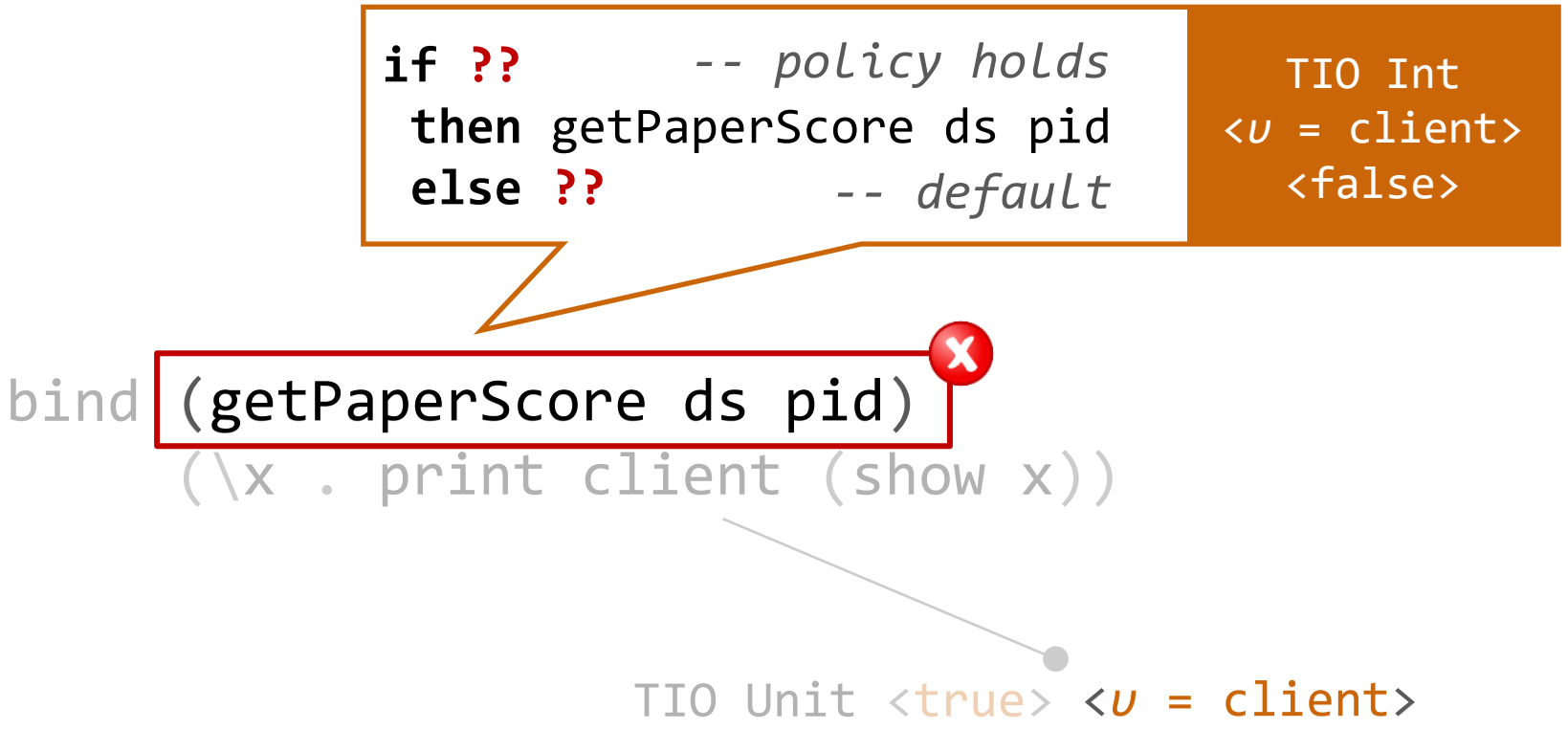
step 3: infer local specification



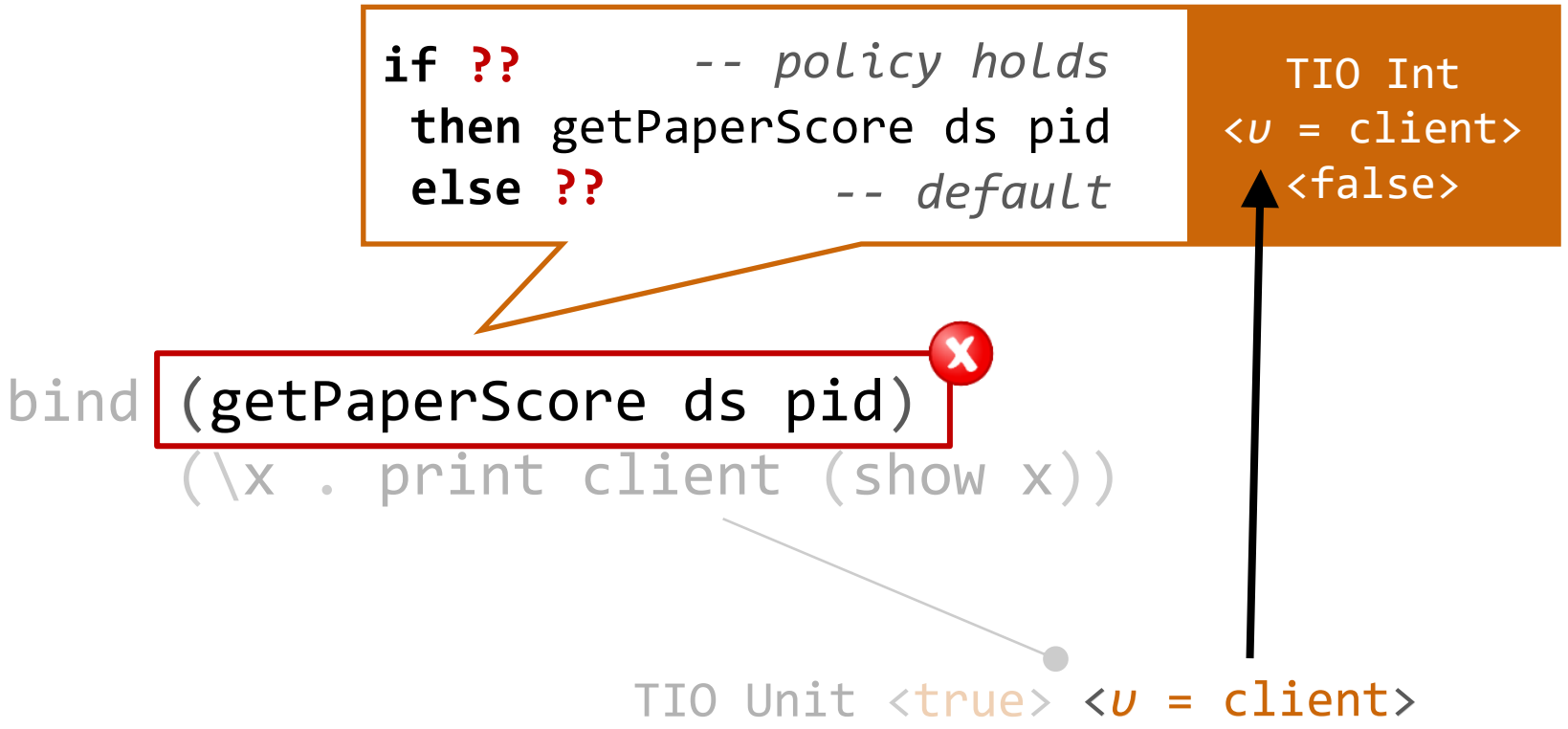
step 3: infer local specification



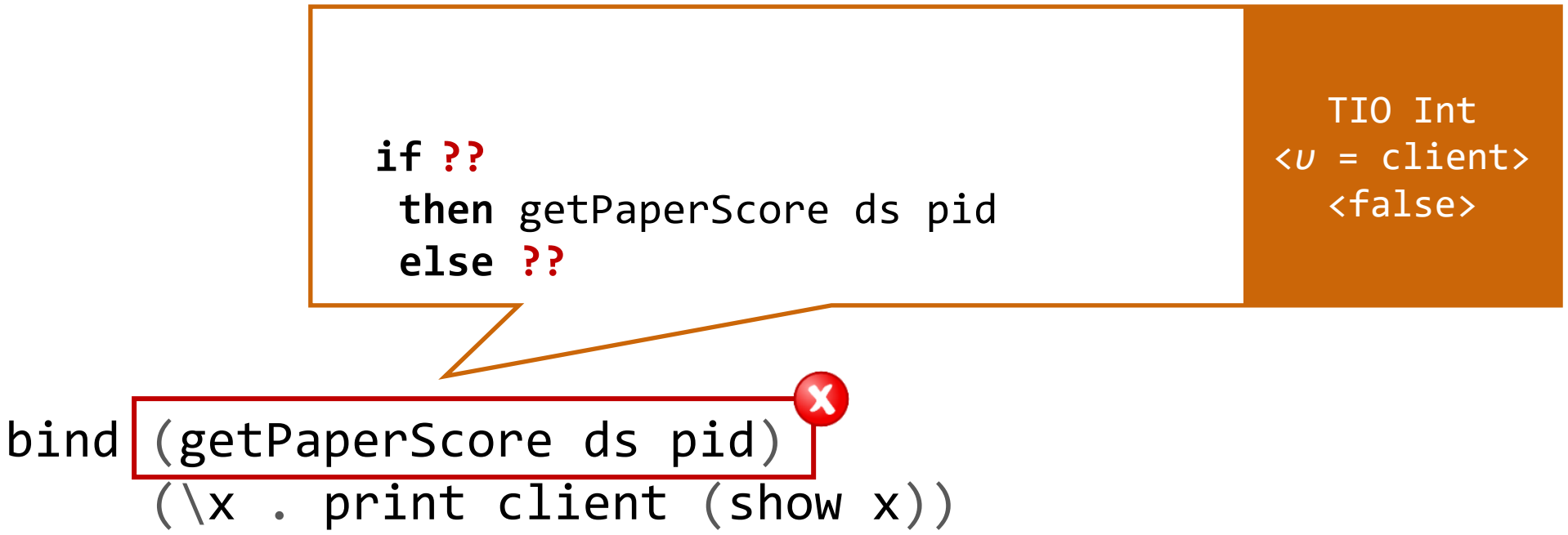
step 3: infer local specification



step 3: infer local specification

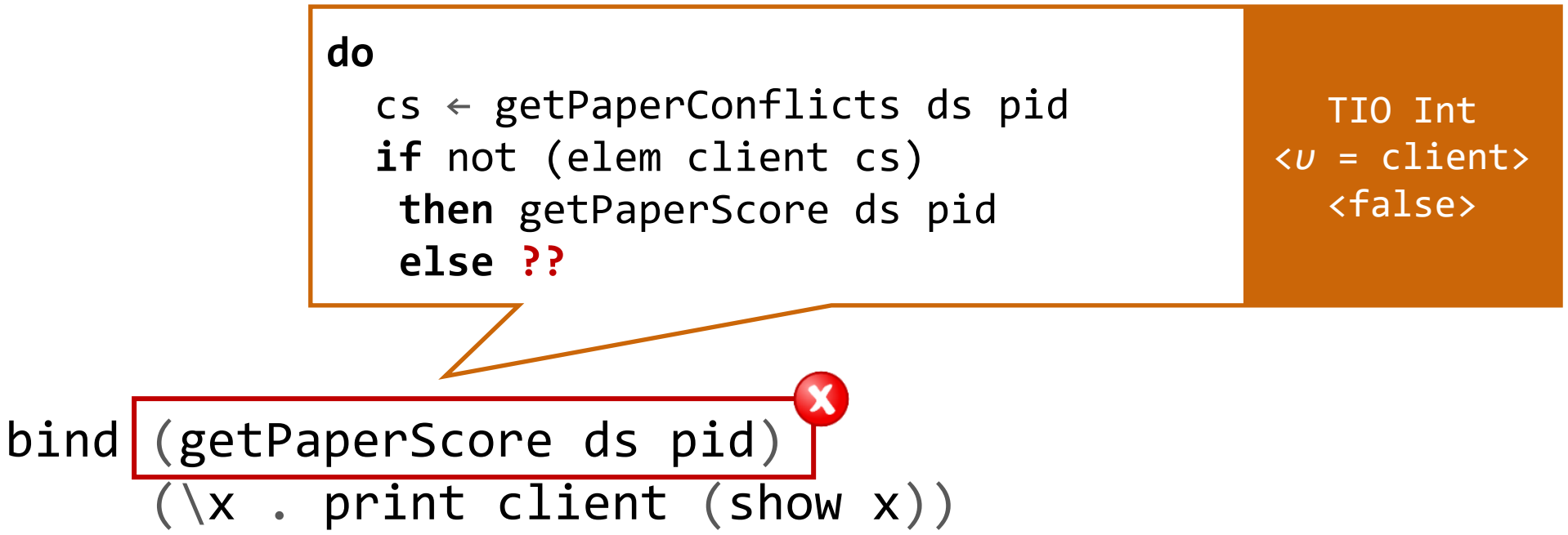


step 4: type-driven synthesis*



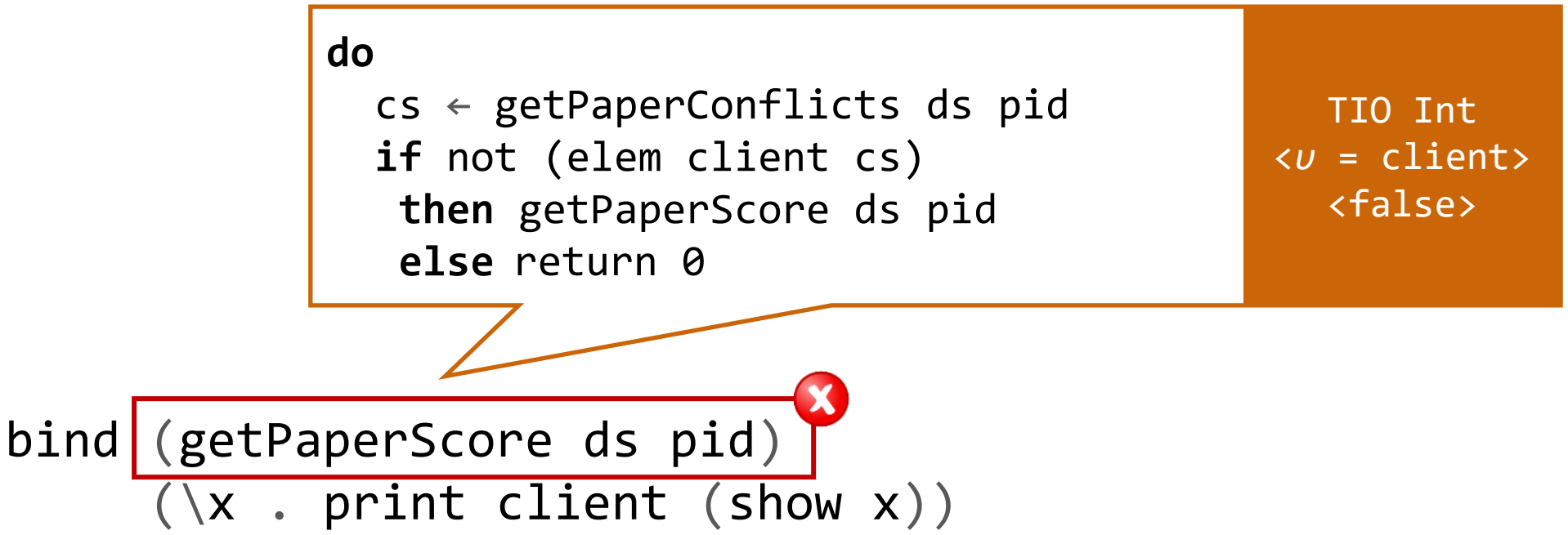
* [Polikarpova, Kuraj, Solar-Lezama. 2016]

step 4: type-driven synthesis*



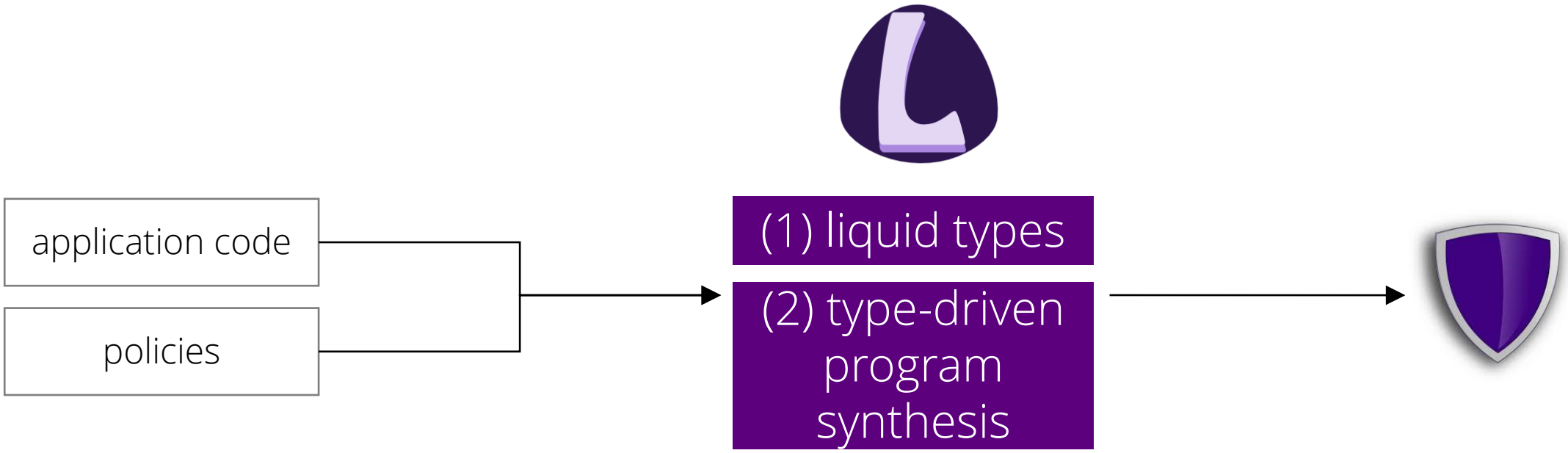
* [Polikarpova, Kuraj, Solar-Lezama. 2016]

step 4: type-driven synthesis*

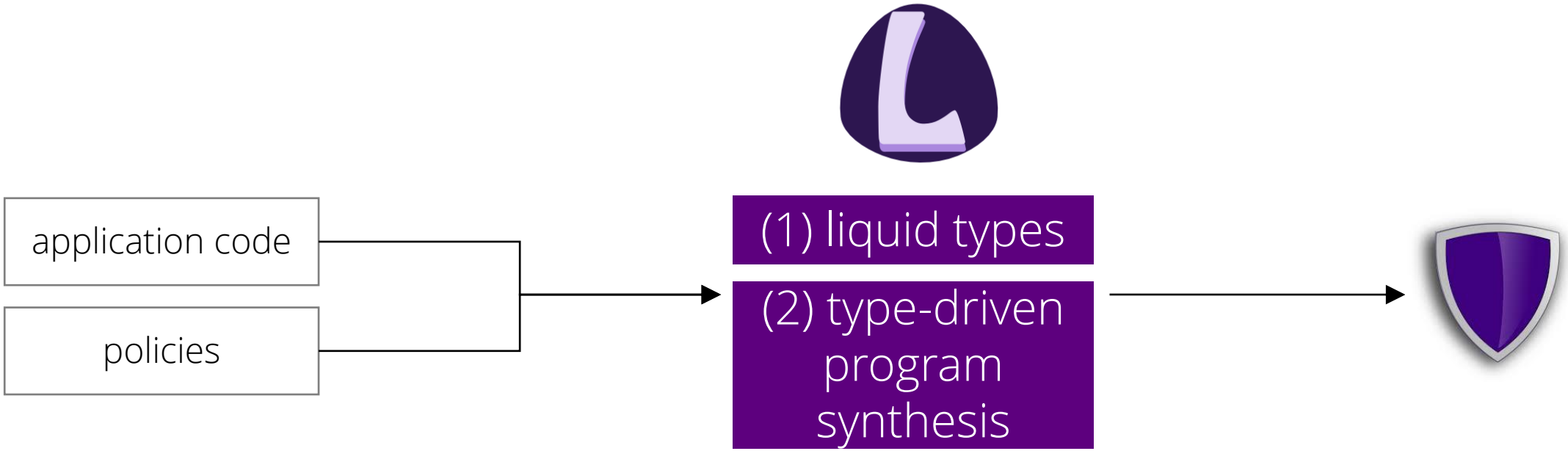


* [Polikarpova, Kuraj, Solar-Lezama. 2016]

lifty


application code
policies
(1) liquid types
(2) type-driven program synthesis

lifty



https://cseweb.ucsd.edu/~npolikarpova/lifty/

ongoing work

STORM

(Security-Typed ORM)

ongoing work

STORM

(Security-Typed ORM)

ongoing work

STORM

(Security-Typed ORM)

\+ ORM

ongoing work

STORM

(Security-Typed ORM)

 + ORM + LiquidHaskell

ongoing work

STORM

(Security-Typed ORM)

\+ ORM +

= secure web framework for Haskell