# Access Permission Contracts for Scripting Languages

## POPL 2012

UNI FREIBURG

Albert-Ludwigs-Universität Freiburg

Phillip Heidegger, Annette Bieniusa, Peter Thiemann

Institut für Informatik
Georges-Köhler-Allee 079
79110 Freiburg

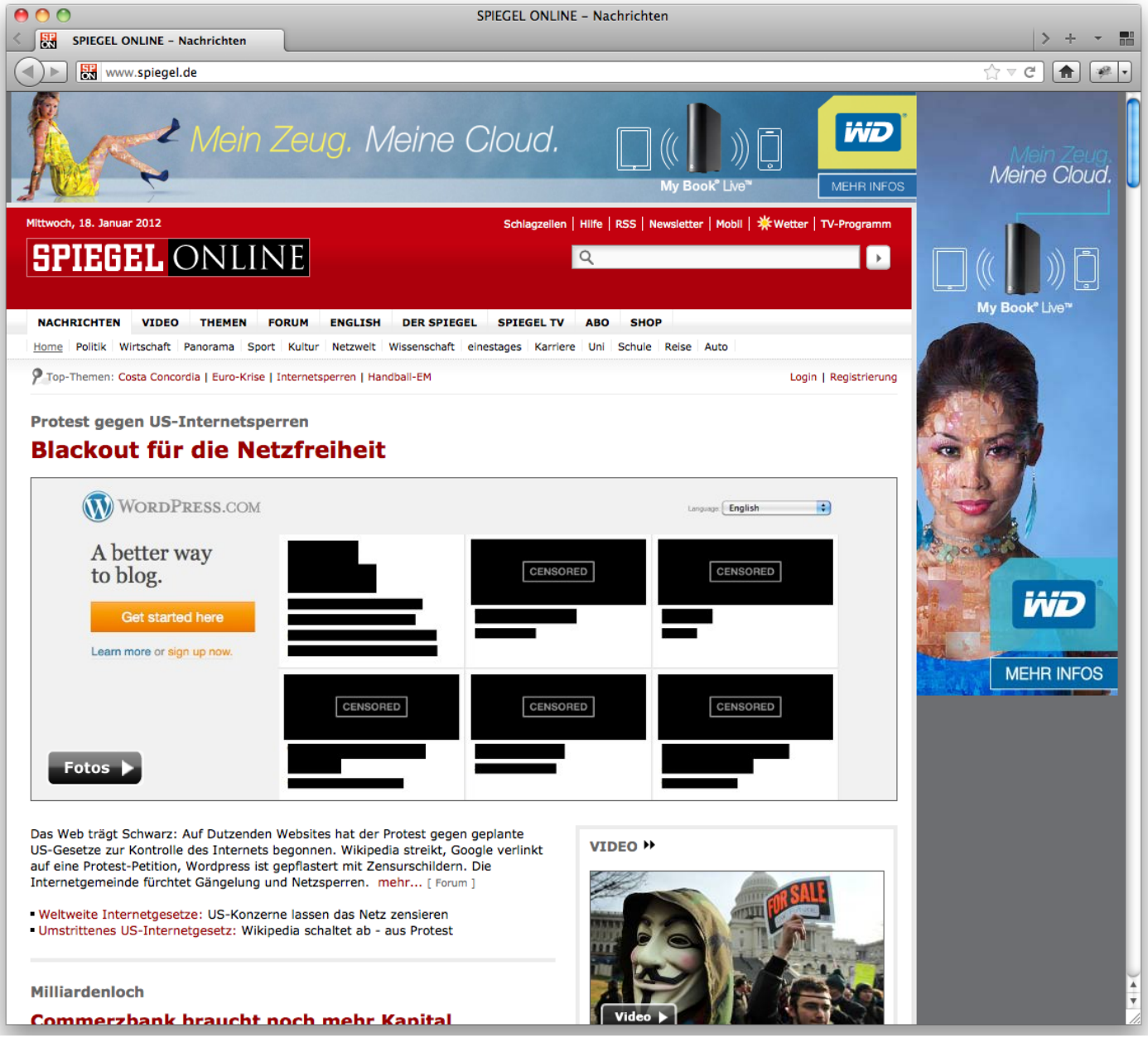
heidegger@informatik.uni-freiburg.de

25. January 2012

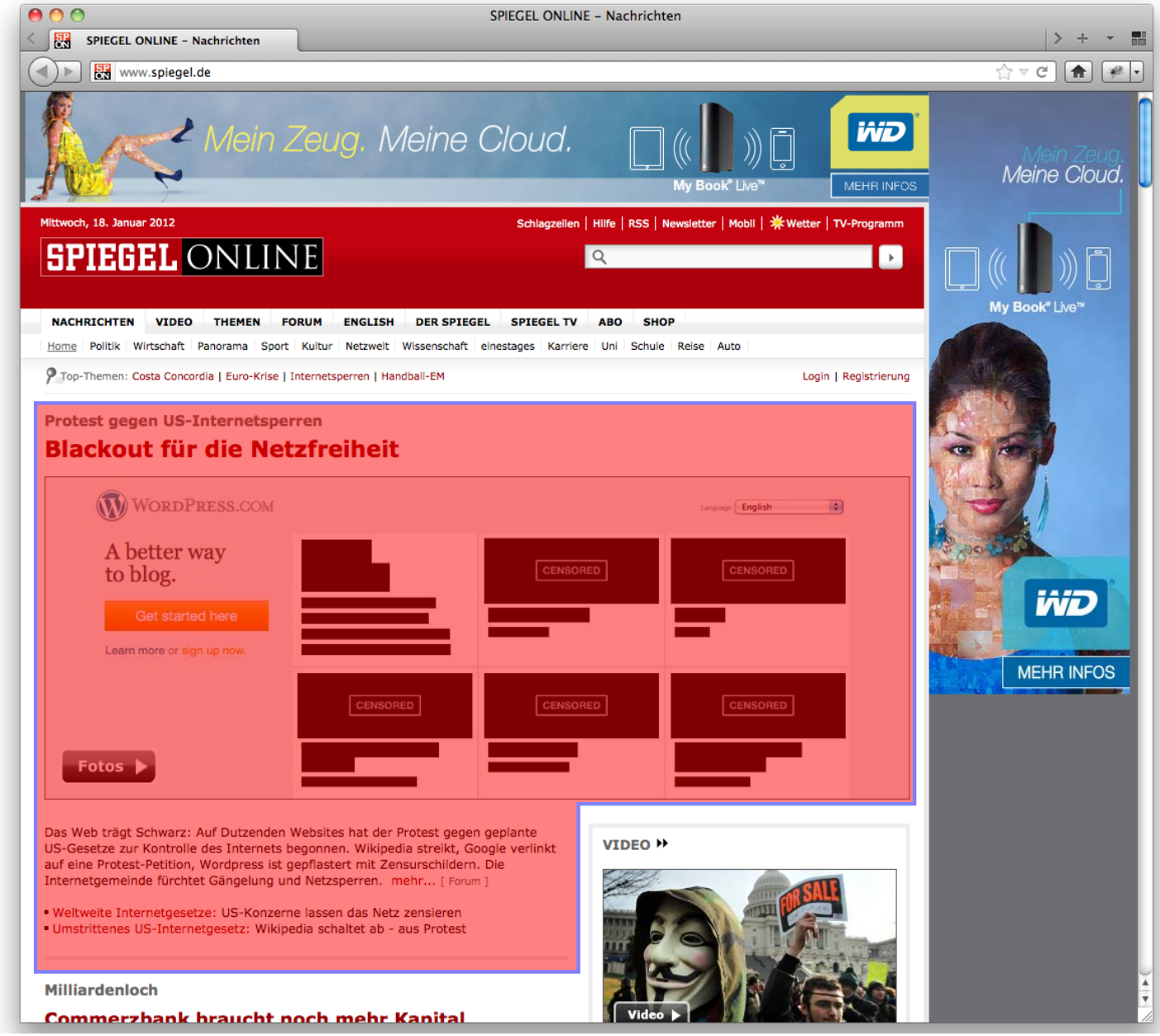
# Motivation


according to: `http://w3techs.com/`, 01/17/12

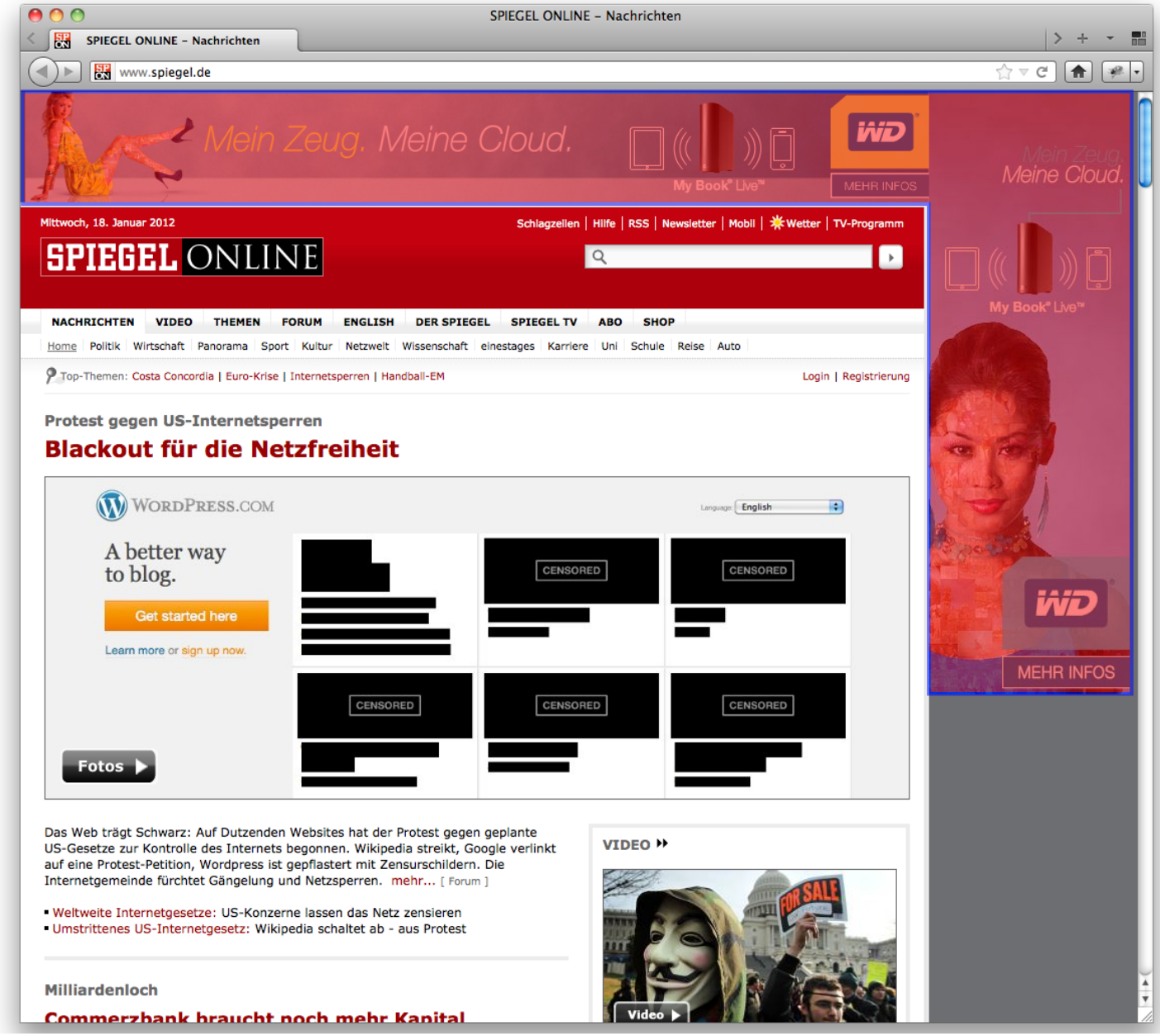
# Situation of a Webpage Programmer



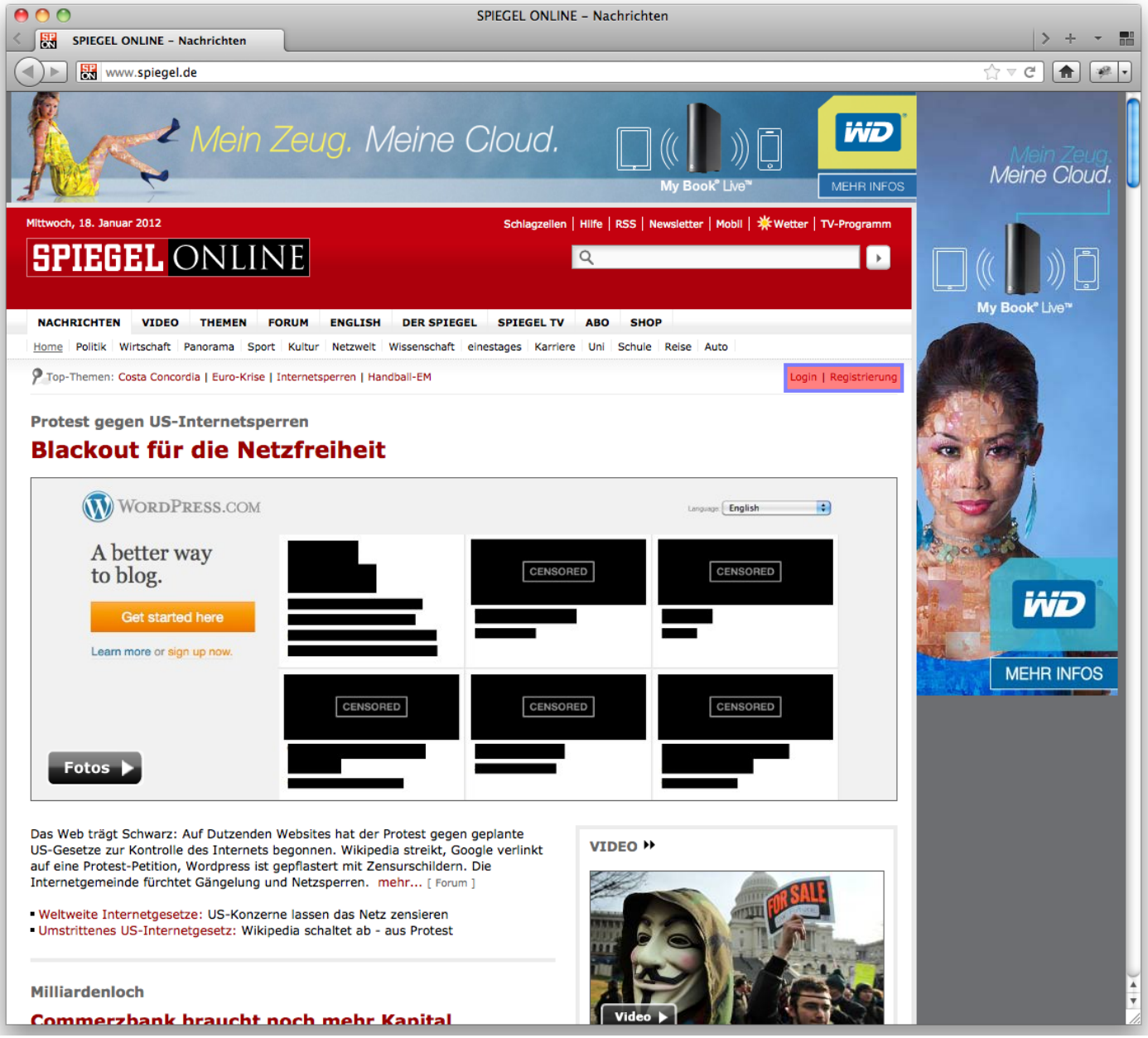
# Situation of a Webpage Programmer



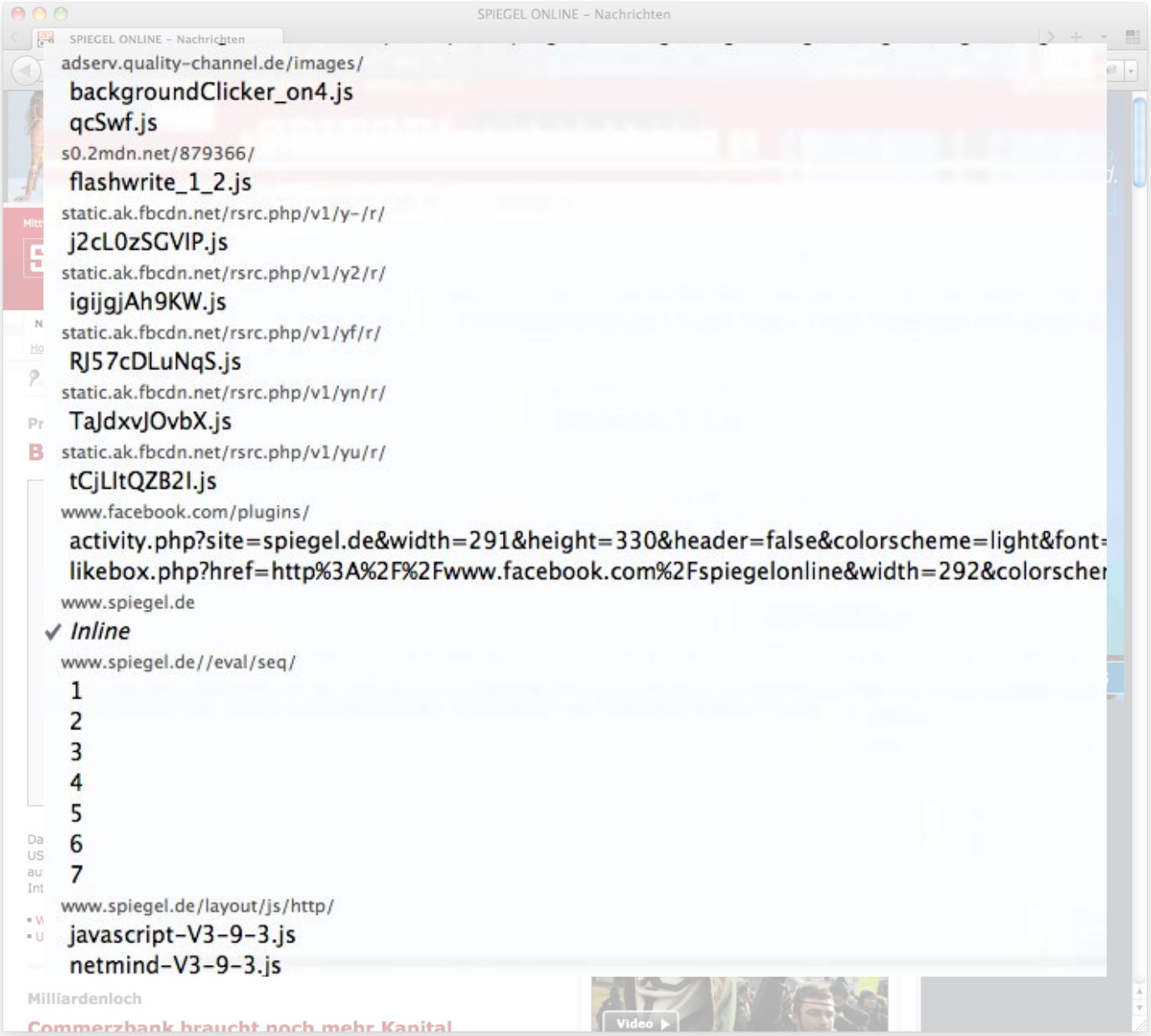
# Situation of a Webpage Programmer



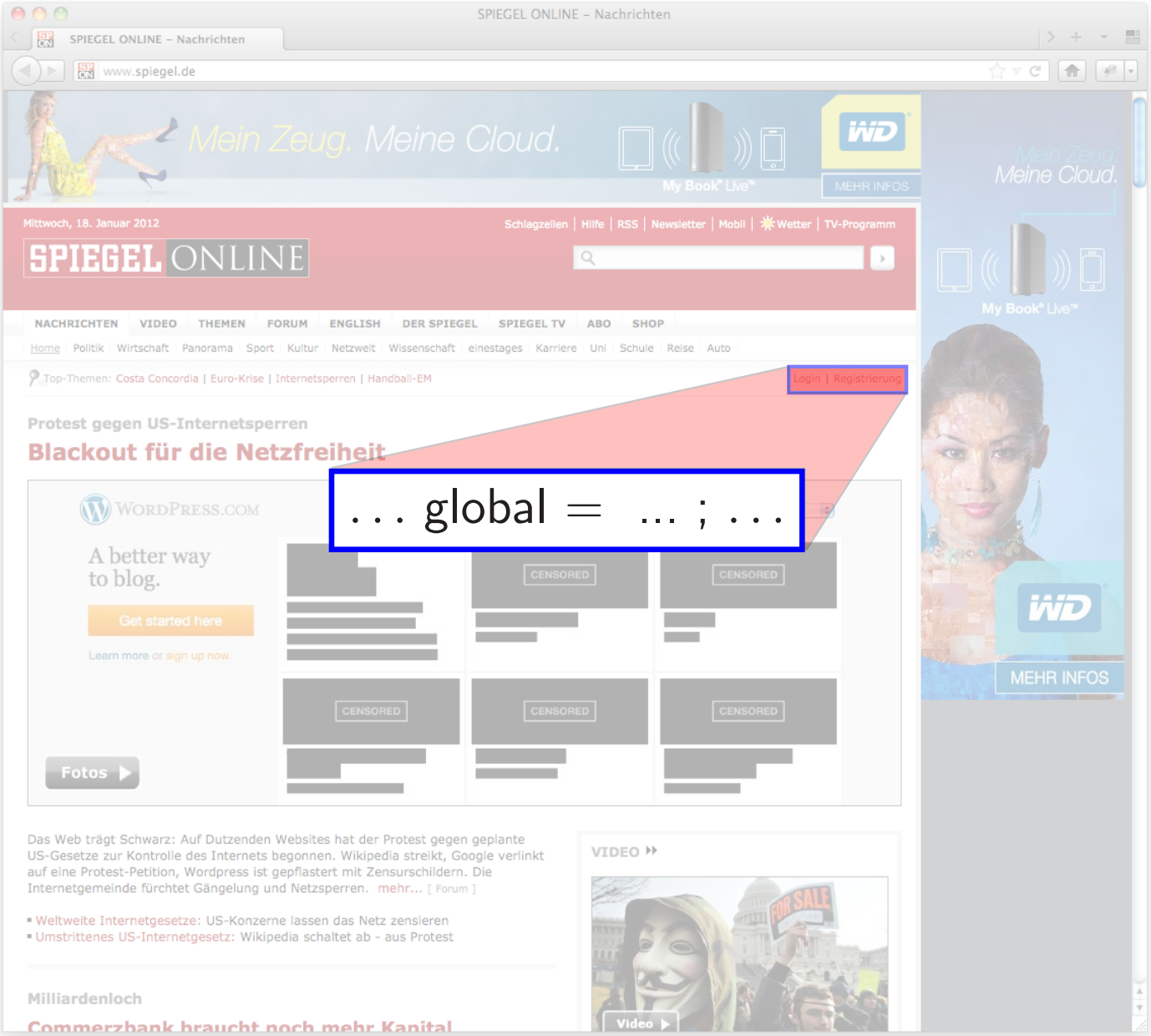
# Situation of a Webpage Programmer



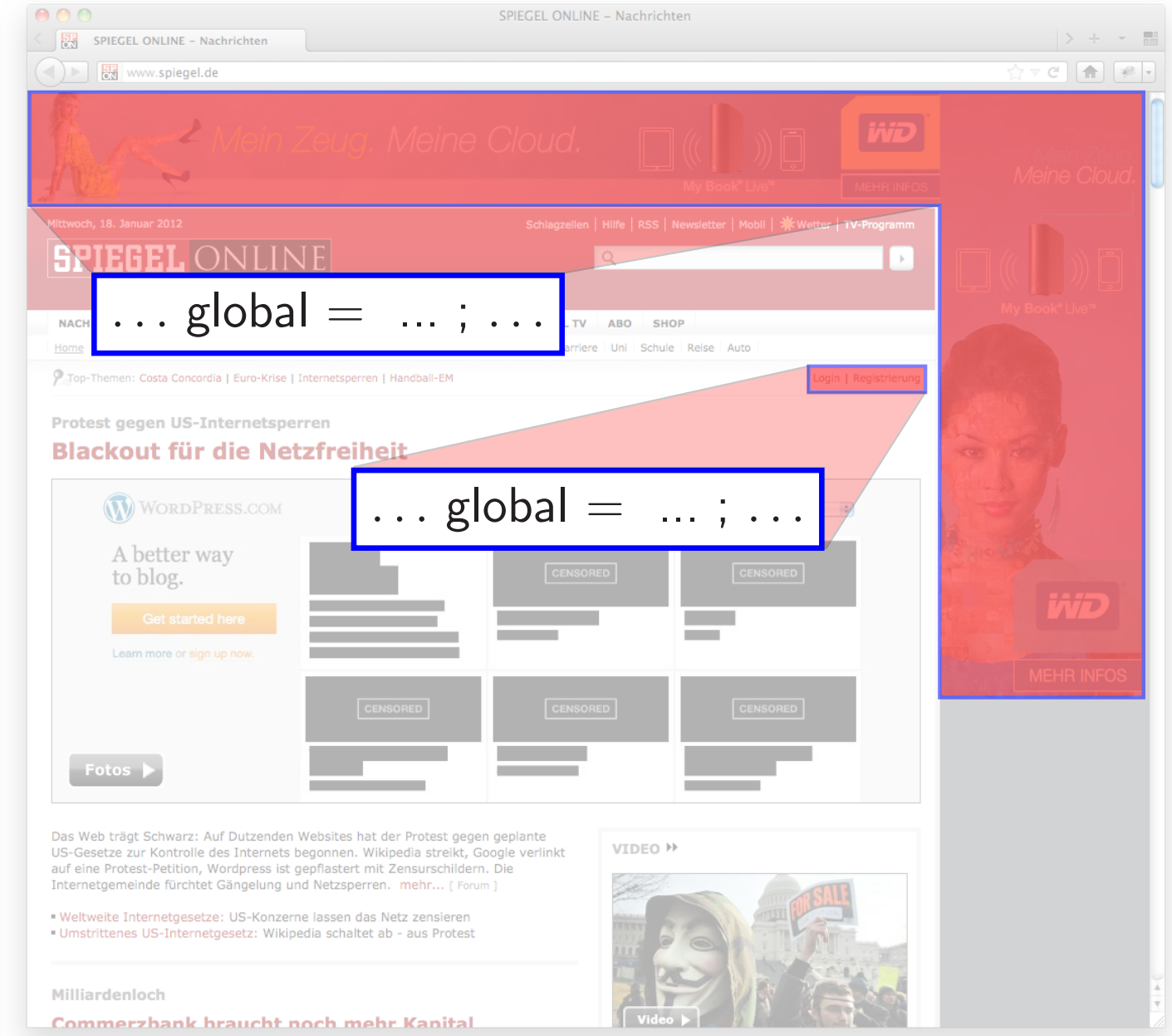
# Situation of a Webpage Programmer



# Situation of a Webpage Programmer



# Situation of a Webpage Programmer



# Problem

UNI FREIBURG

## Problem

- Uncontrolled side effects may cause unexpected behavior
- Code is aggregated by dynamic loading

Maintenance is a nightmare!

# Problem

UNI FREIBURG

## Problem

- Uncontrolled side effects may cause unexpected behavior
- Code is aggregated by dynamic loading

Maintenance is a nightmare!

## What can we do?

- static approach?
- documentation?
- monitoring!

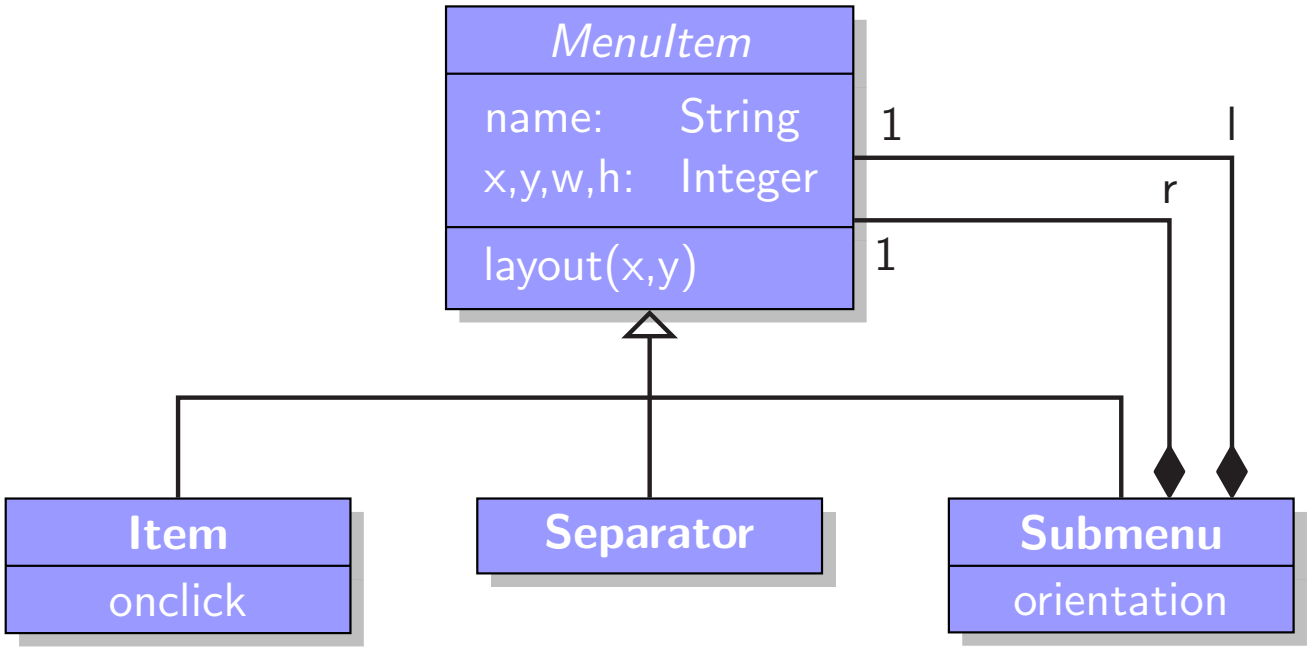
# JSConTest: A tool for partial specifications

UNI FREIBURG

## Overview

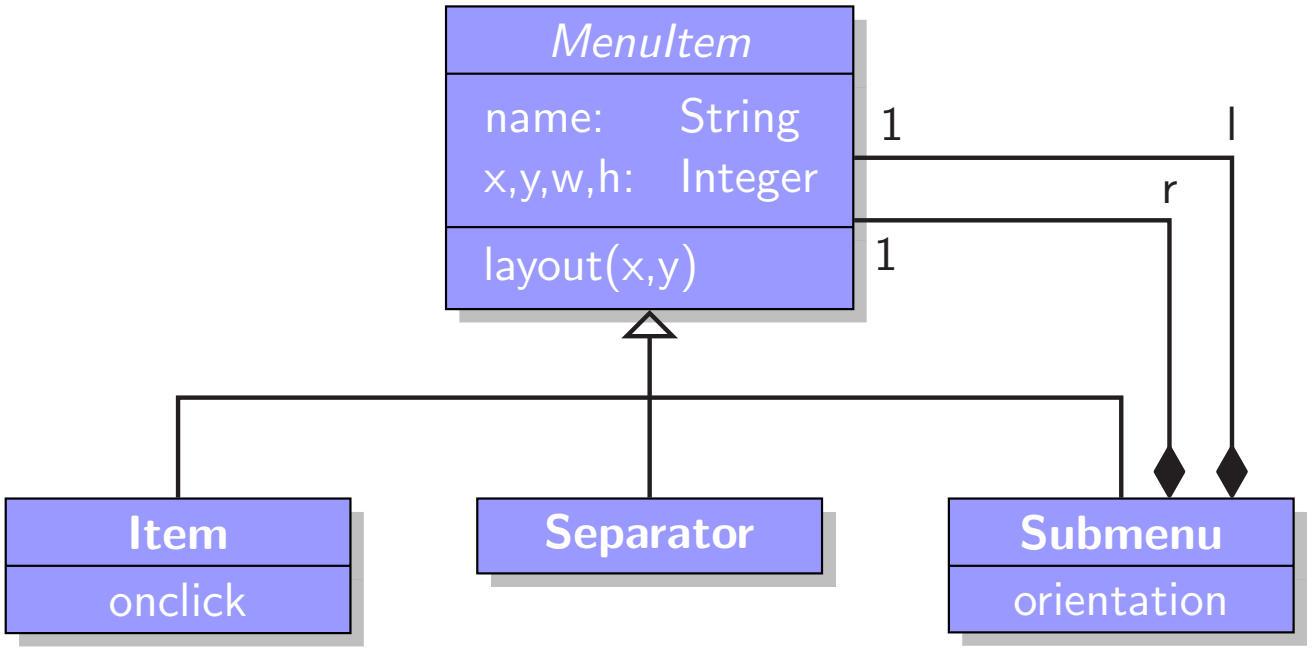
- Type contracts
  - → describe the *functional behavior* of an operation
- Access permission contracts
  - → describe the *side effect* of an operation

# Example – Menu



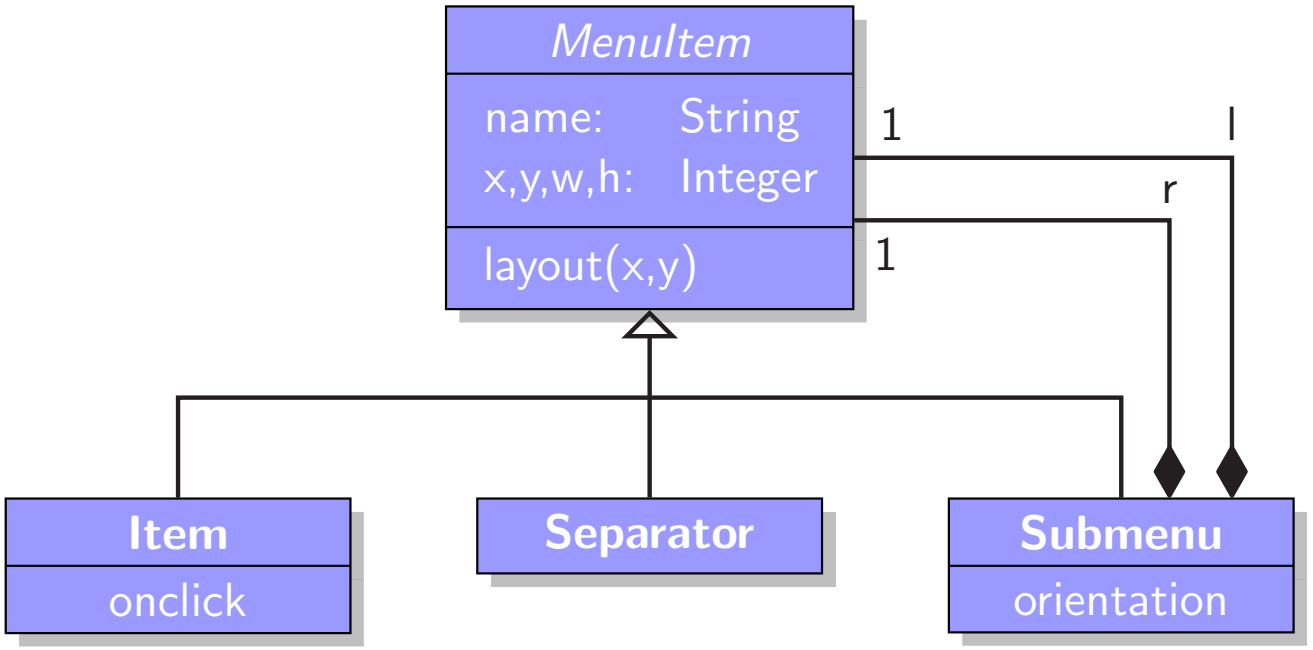

```
/*c (int, int) -> {w:int,h:int}      ← type signature
     with [this./l|r/*./x|y|w|h/] ← access permission
*/
function layout(x,y) { ... }
```

# Example – Menu




```
/*c (int, int) -> {w:int,h:int}      ← type signature
     with [this./l|r/*./x|y|w|h/] ← access permission
*/
function layout(x,y) { ... }
```

# Example – Menu

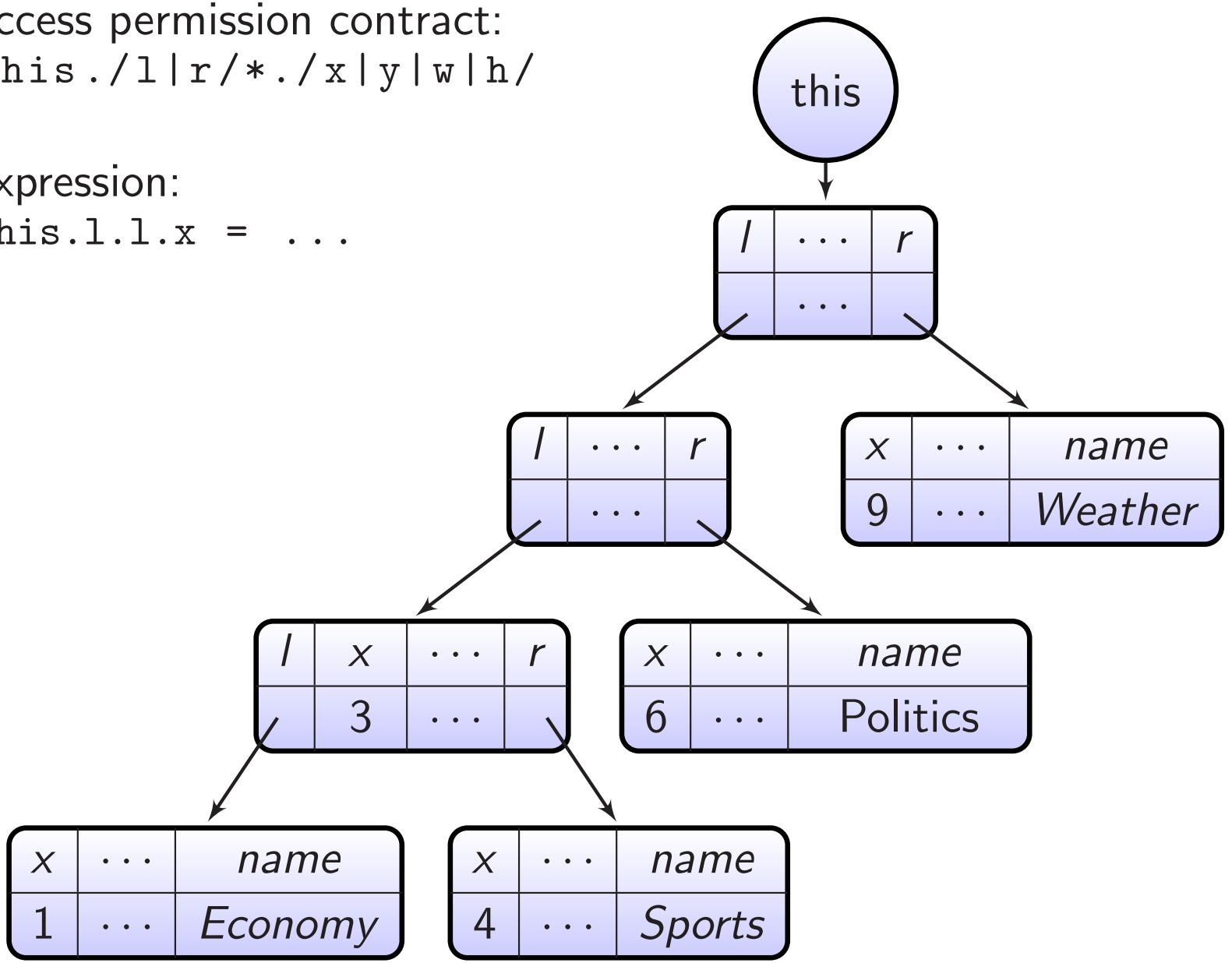



```
/*c (int, int) -> {w:int,h:int}      ← type signature
     with [this./l|r/*./x|y|w|h/] ← access permission
*/
function layout(x,y) { ... }
```

# Example – Menu

access permission contract:
`this./l|r/*./x|y|w|h/`

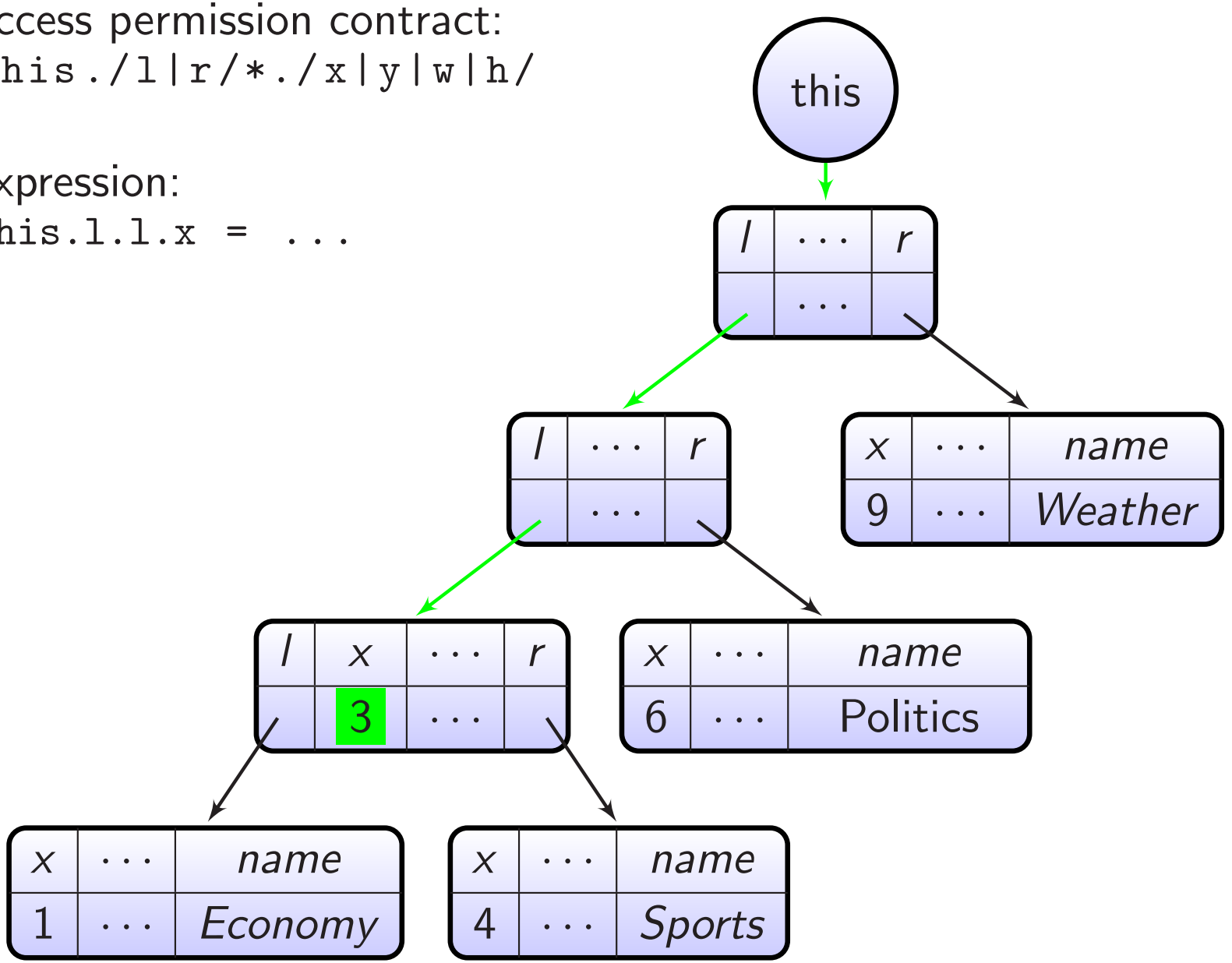
expression:
`this.l.l.x = ...`



# Example – Menu

access permission contract:
`this./l|r/*./x|y|w|h/`

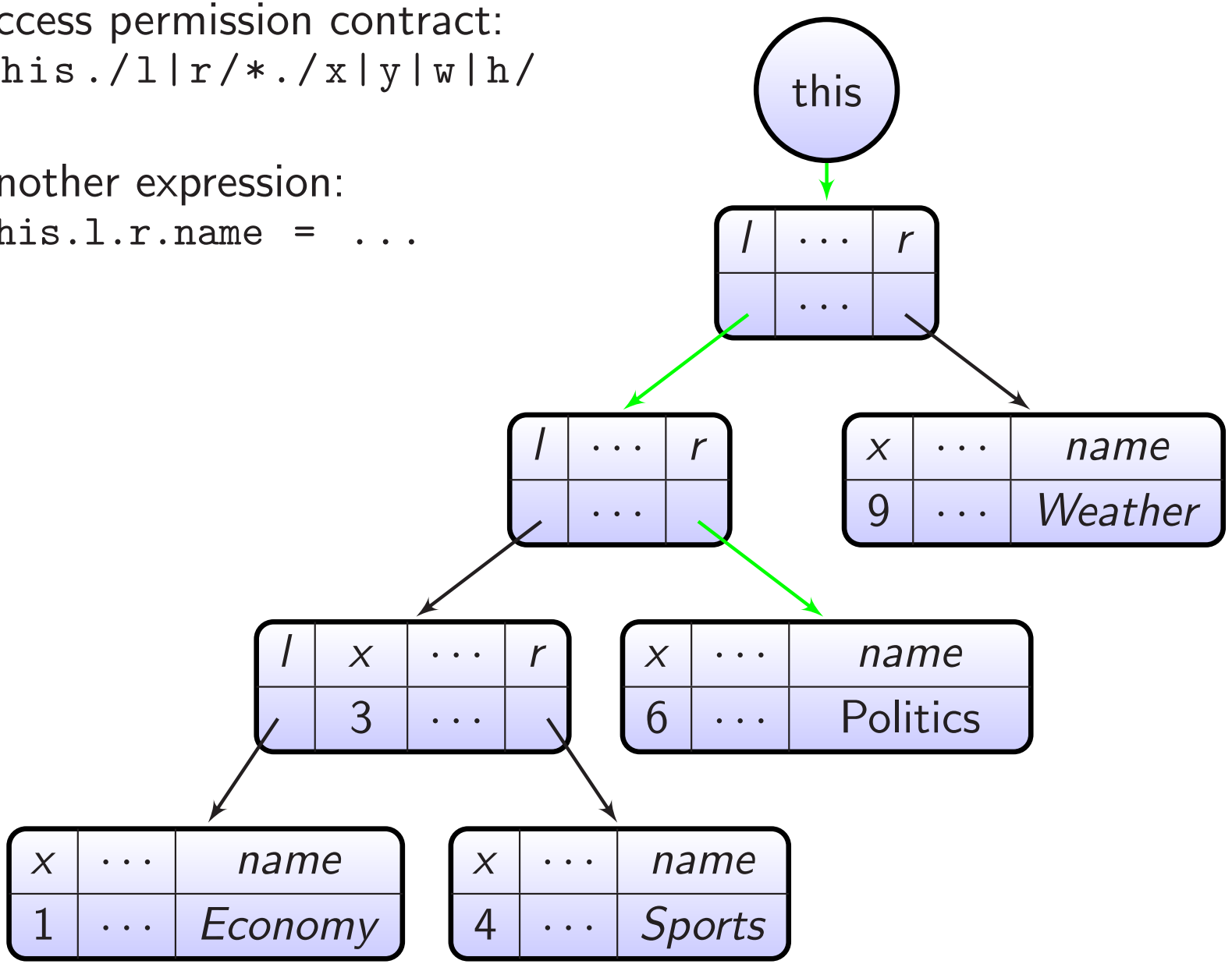
expression:
`this.l.l.x = ...`



# Example – Menu

access permission contract:
`this./l|r/*./x|y|w|h/`

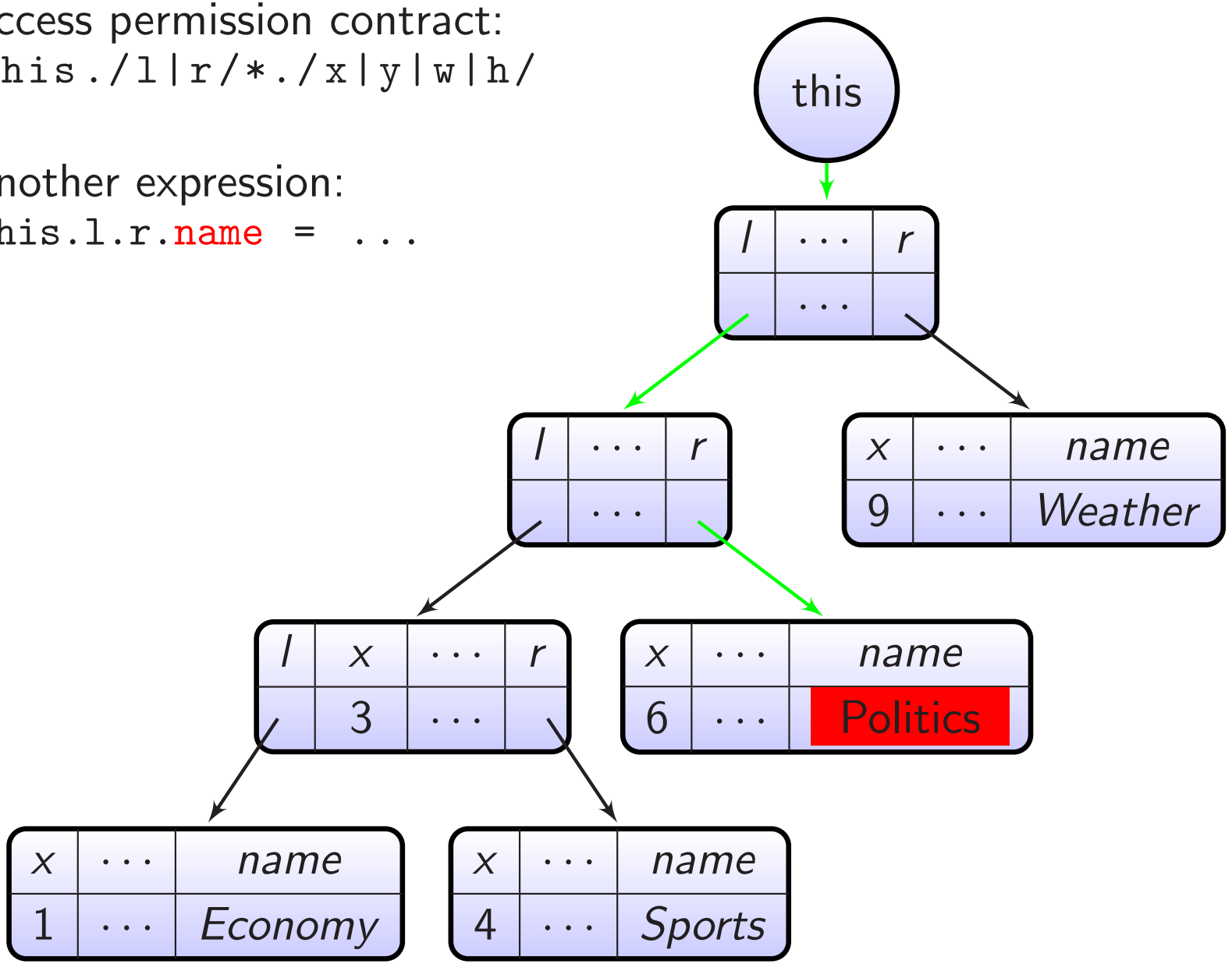
another expression:
`this.l.r.name = ...`



# Example – Menu

access permission contract:
`this./l|r/*./x|y|w|h/`

another expression:
`this.l.r.name = ...`



# Semantics of Access Permission Contracts

UNI FREIBURG

Is the meaning of an

access permission settled?

# Semantics of Access Permission Contracts

UNI FREIBURG

Is the meaning of an

access permission settled?

NO!

There are three open issues...

# Syntactic Interpretation – Example

UNI FREIBURG

```
/*c (obj,obj) → any with [x.a, y.a, y.a.b] */
function b(x,y) {
  ...
  ... y.a.b ... // violation?
  ...
}
```

# Syntactic Interpretation – Example

UNI FREIBURG

```
/*c (obj,obj) → any with [x.a, y.a, y.a.b] */
function b(x,y) {
  var tmp = y.a;
  y.a = x.a;
  y.a.b = 42; // violation?
  y.a = tmp;
}
```

# Syntactic Interpretation – Example

UNI FREIBURG

```
/*c (obj,obj) → any with [x.a, y.a, y.a.b] */
function b(x,y) {
  var tmp = y.a;
  y.a = x.a;
  y.a.b = 42; // violation?  YES, please!!
  y.a = tmp;
}
```

## From the view of the caller

- The function modifies `x.a.b`.
- By introducing local aliases, a function may execute side effects that are not allowed

# Syntactic Interpretation vs. Pre-State Snapshot

UNI FREIBURG

## ~~Syntactic Interpretation~~

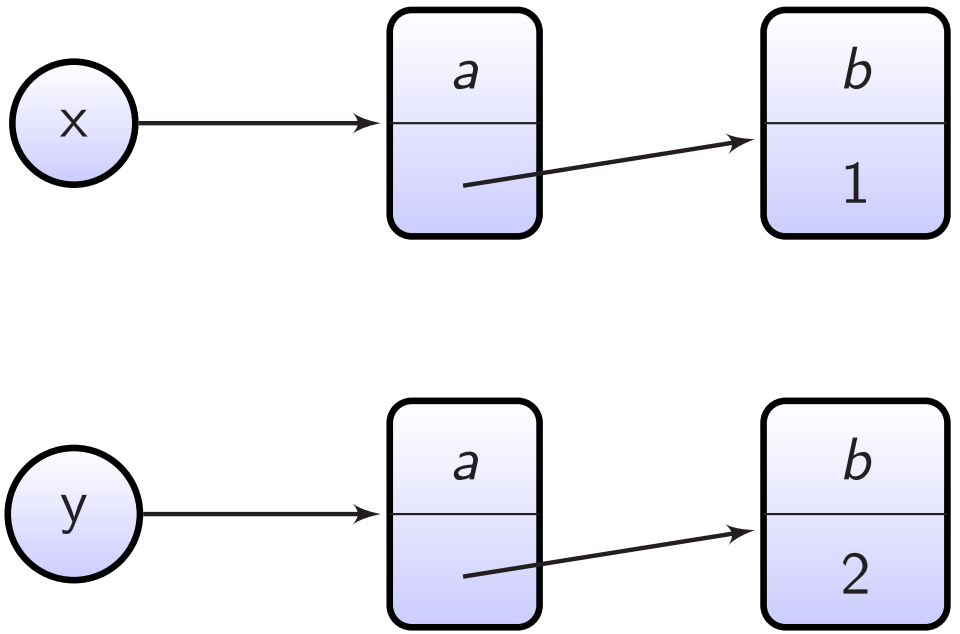
An access permission contract is interpreted syntacticly.

## Pre-State Snapshot

An access permission contract is interpreted with respect to the heap at the time the contract is installed.

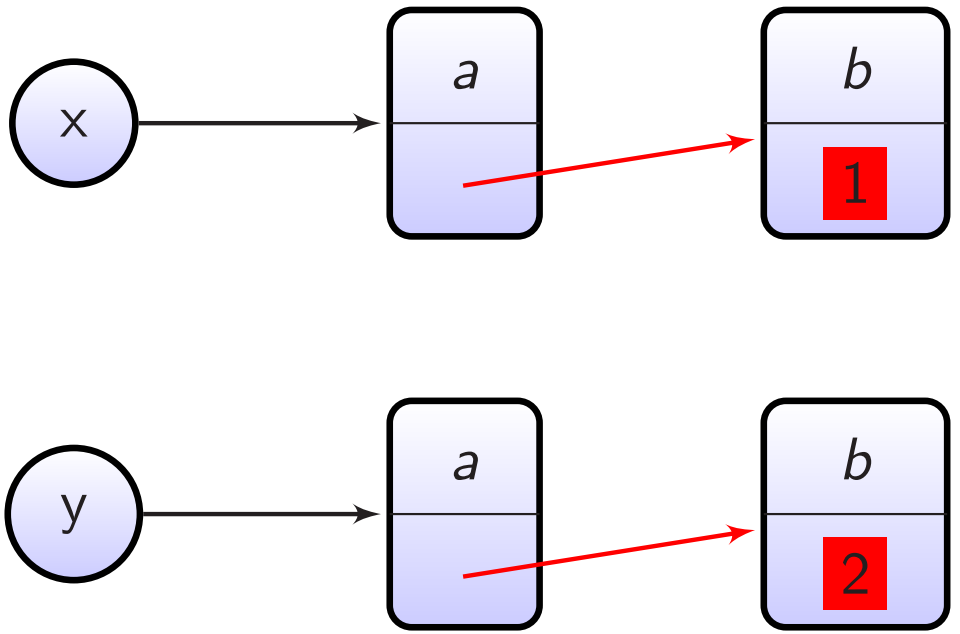
# Pre-State Snapshot – Example

```
/*c (obj,obj) → any with [x.a , y.a , y.a.b ] */
function b(x,y) {
        y.a = x.a;
        y.a.b = 42; // violation?
}
```



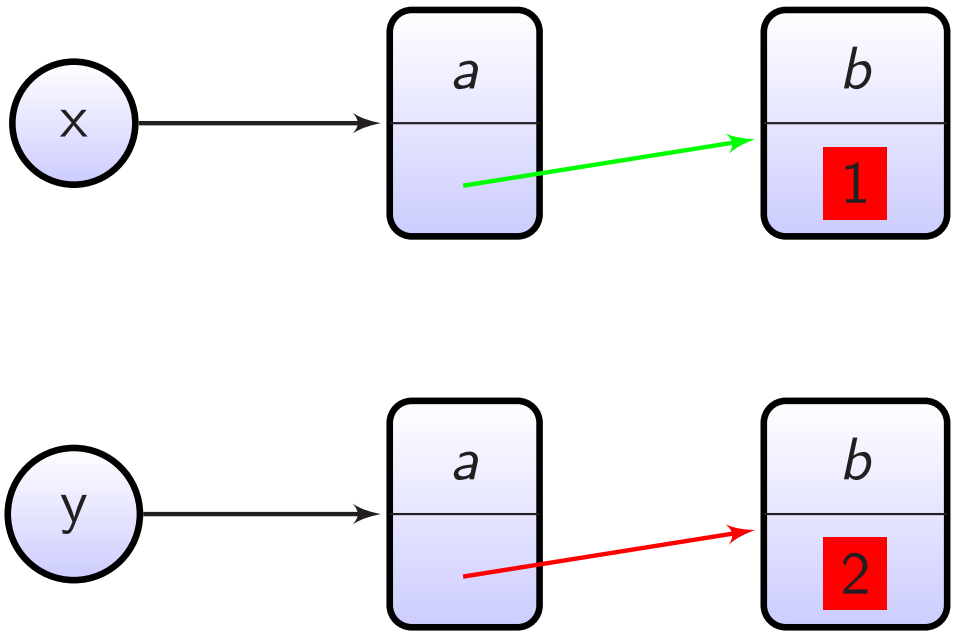
# Pre-State Snapshot – Example

```
/*c (obj,obj) → any with [x.a , y.a , y.a.b ] */
function b(x,y) {
        y.a = x.a;
        y.a.b = 42; // violation?
}
```



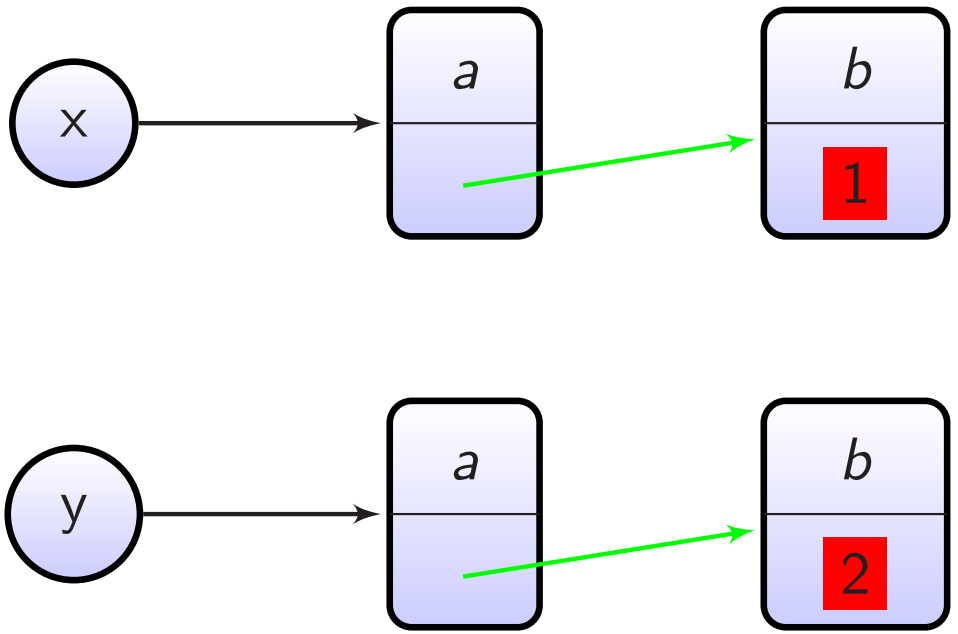
# Pre-State Snapshot – Example

```
/*c (obj,obj) → any with [x.a , y.a , y.a.b ] */
function b(x,y) {
        y.a = x.a;
        y.a.b = 42; // violation?
}
```



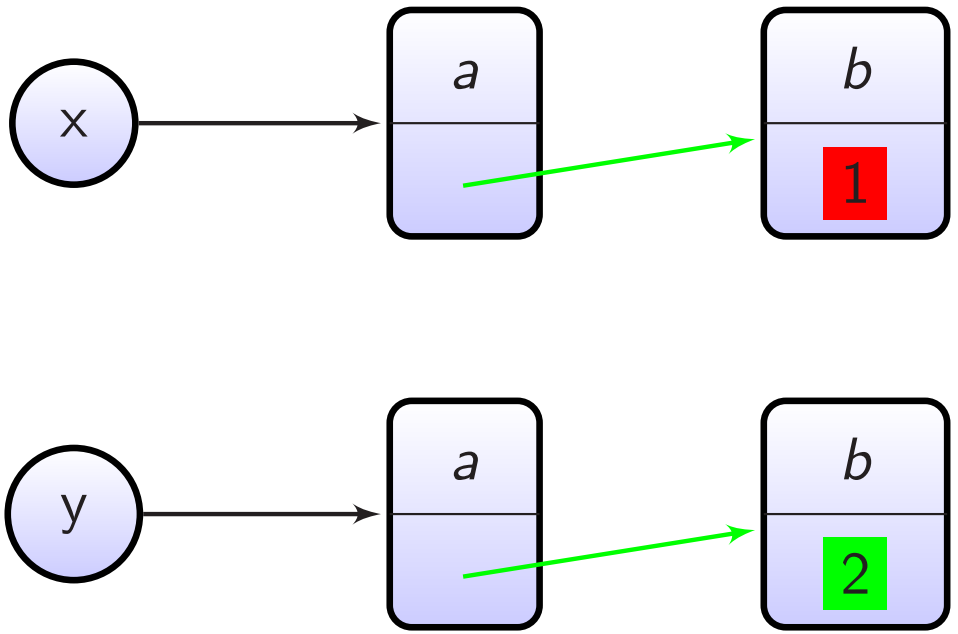
# Pre-State Snapshot – Example

```
/*c (obj,obj) → any with [x.a , y.a , y.a.b ] */
function b(x,y) {
        y.a = x.a;
        y.a.b = 42; // violation?
}
```



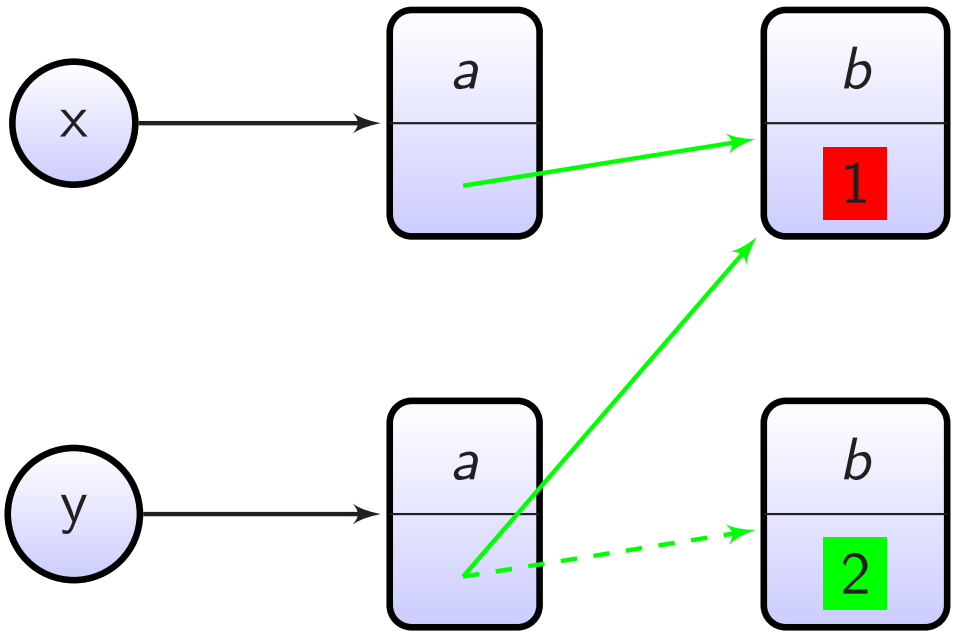
# Pre-State Snapshot – Example

```
/*c (obj,obj) → any with [x.a , y.a , y.a.b ] */
function b(x,y) {
 →      y.a = x.a;
        y.a.b = 42; // violation?
}
```



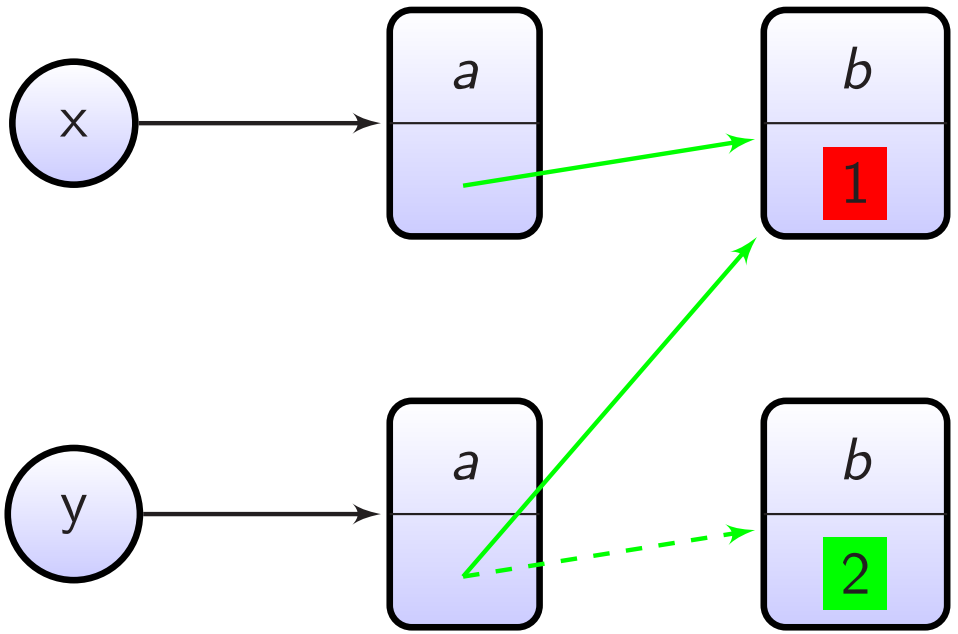
# Pre-State Snapshot – Example

```
/*c (obj,obj) → any with [x.a , y.a , y.a.b ] */
function b(x,y) {
        y.a = x.a;
 →      y.a.b = 42; // violation?
}
```



# Pre-State Snapshot – Example

```
/*c (obj,obj) → any with [x.a , y.a , y.a.b ] */
function b(x,y) {
        y.a = x.a;
        y.a.b = 42; // violation?  YES
}
```



# Design Principles

UNI FREIBURG

- Pre-State Snapshot
- ...

# Example – Dynamic Extent vs. Static Extent

UNI FREIBURG

```
/*c (obj) → any with [x.a] */
function d1(x) {
  x.a = 1;     // violation if called from d2?
}

/*c (obj) → any with [] */
function d2(x) {
  return d1(x);
}
```

# Dynamic Extent vs. Static Extent

UNI FREIBURG

## Dynamic Extent

An access permission for a function is in force for the duration of the function activation.

## ~~Static Extent~~

An access permission for a function is in force in its function body.

# Design Principles

UNI FREIBURG

- Pre-State Snapshot
- Dynamic Extent
- ...

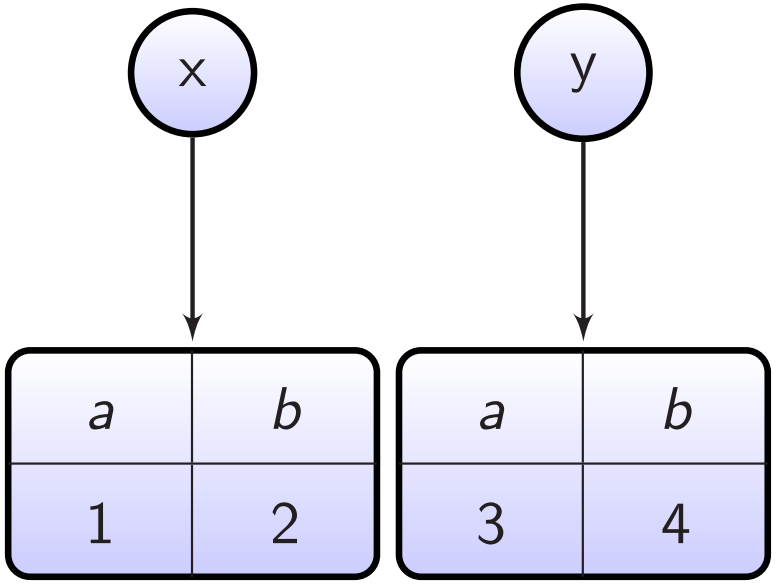
# Location Based Interpretation – Example

UNI FREIBURG

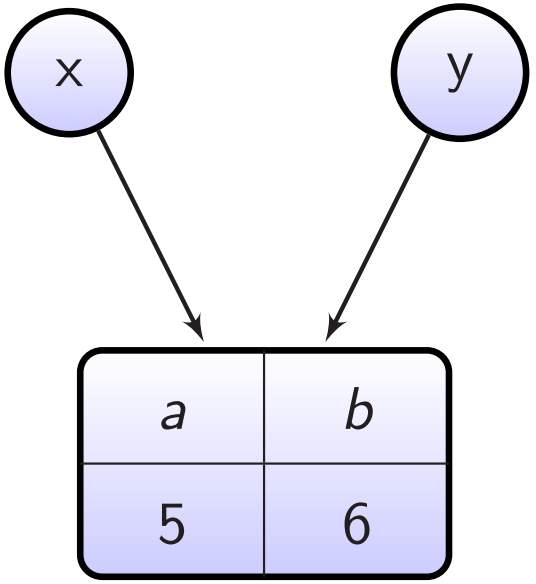
```
/*c (obj,obj) → any with [ x.b, y.a ] */
function h(x,y) {
  y.a = 1;
  y.b = 2;   // violation?
}
```

# Location Based Interpretation – Example

```
/*c (obj,obj) → any with [ x.b, y.a ] */
function h(x,y) {
  y.a = 1;
  y.b = 2;   // violation?
}
```



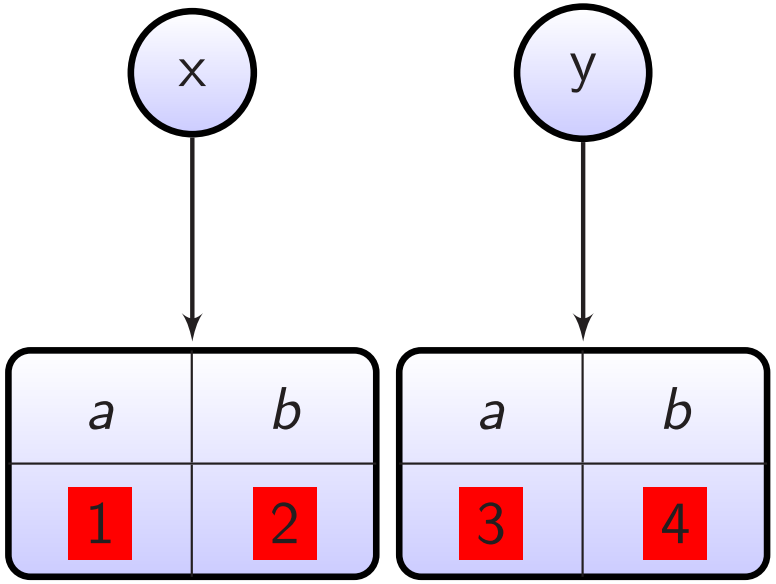

```
h({a:1,b:2},{a:3,b:4});
```



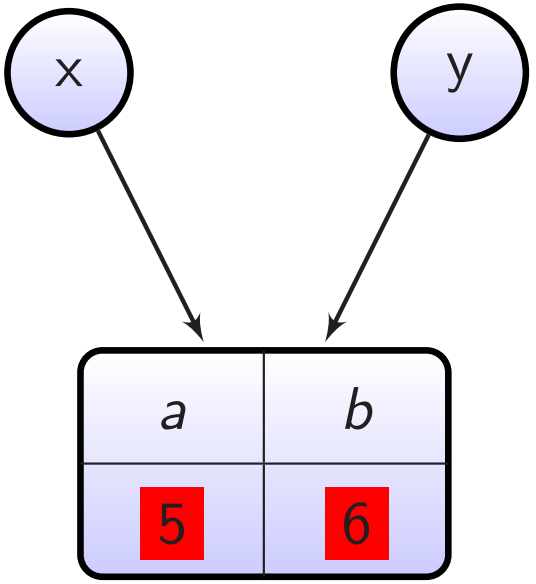

```
var o = {a:5,b:6};
h(o,o);
```

# Location Based Interpretation – Example

```
/*c (obj,obj) → any with [ x.b, y.a ] */
function h(x,y) {
  y.a = 1;
  y.b = 2;   // violation?
}
```



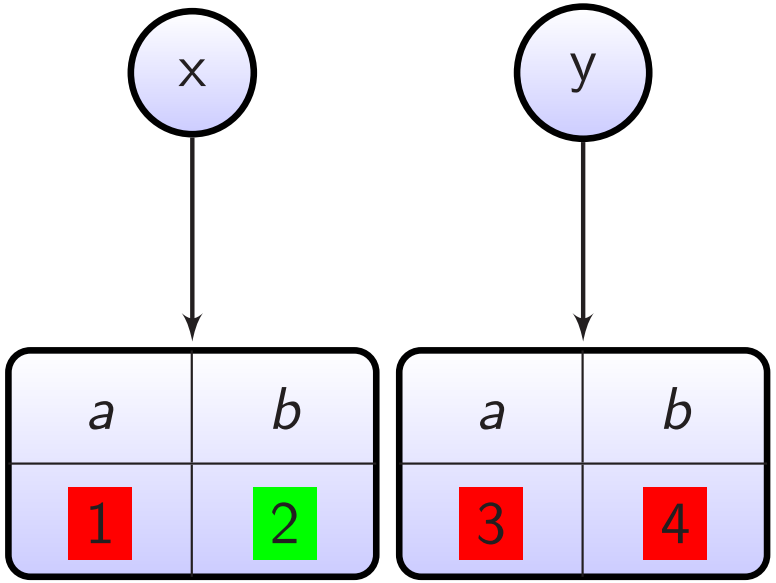

```
h({a:1,b:2},{a:3,b:4});
```



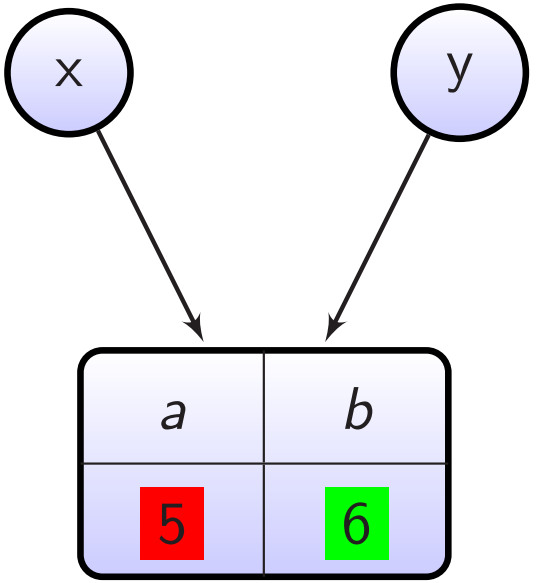

```
var o = {a:5,b:6};
h(o,o);
```

# Location Based Interpretation – Example

```
/*c (obj,obj) → any with [ x.b, y.a ] */
function h(x,y) {
  y.a = 1;
  y.b = 2;   // violation?
}
```



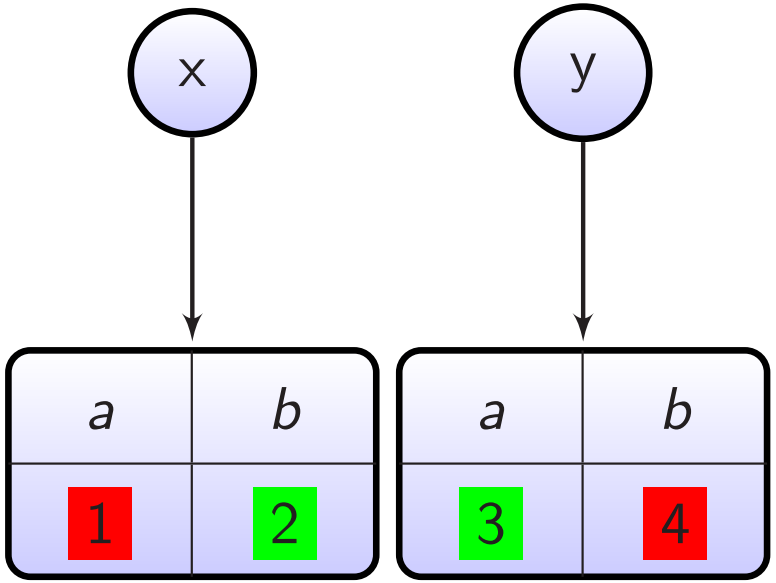

```
h({a:1,b:2},{a:3,b:4});
```



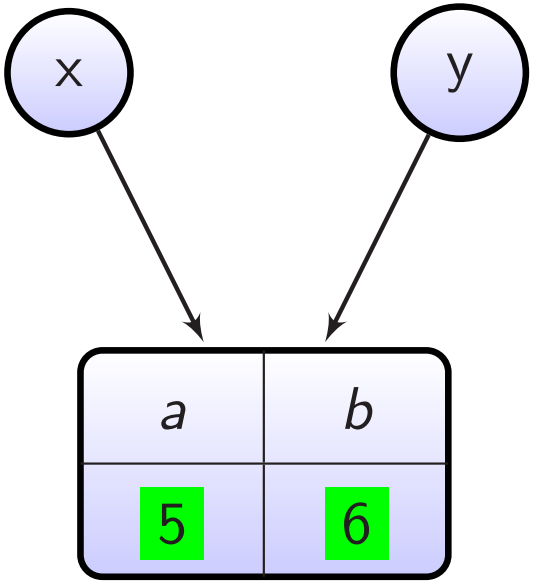

```
var o = {a:5,b:6};
h(o,o);
```

# Location Based Interpretation – Example

```
/*c (obj,obj) → any with [ x.b, y.a ] */
function h(x,y) {
  y.a = 1;
  y.b = 2;   // violation?
}
```



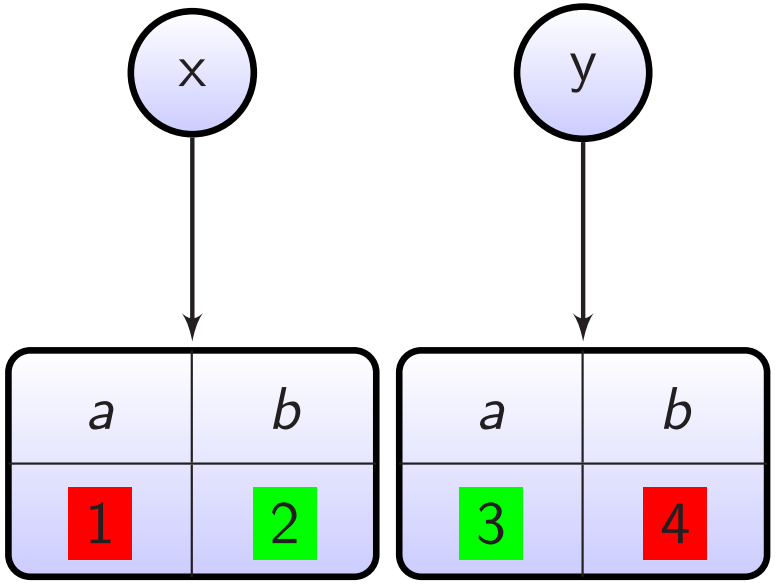

```
h({a:1,b:2},{a:3,b:4});
```



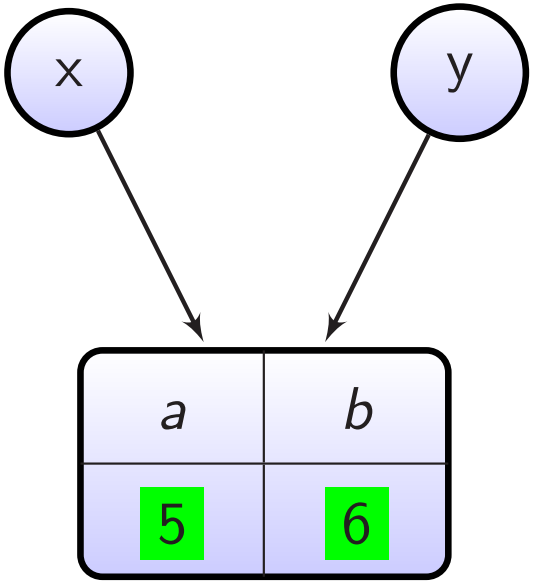

```
var o = {a:5,b:6};
h(o,o);
```

# Location Based Interpretation – Example

```
/*c (obj,obj) → any with [ x.b, y.a ] */
function h(x,y) {
  y.a = 1;
  y.b = 2;   // violation? left: YES, right: NO
}
```




```
h({a:1,b:2},{a:3,b:4});
```




```
var o = {a:5,b:6};
h(o,o);
```

# Location Based Interpretation

UNI FREIBURG

## Conclusion

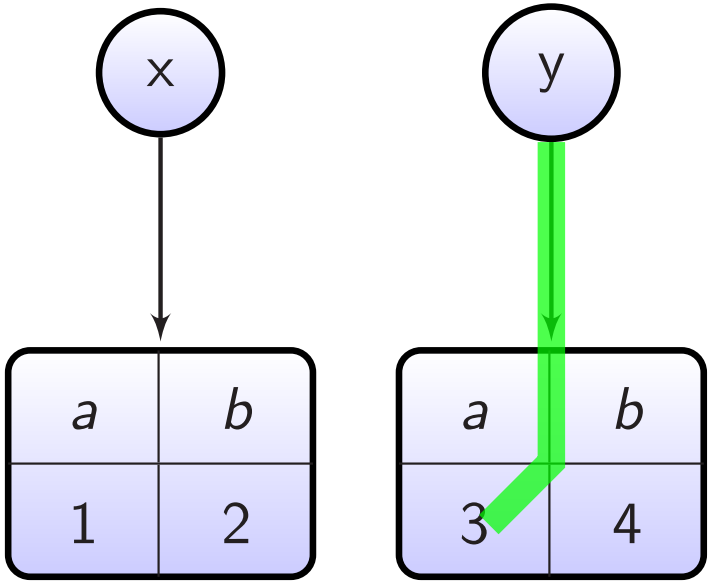
Violation occurs depending on the aliases of the parameters

## Requirement

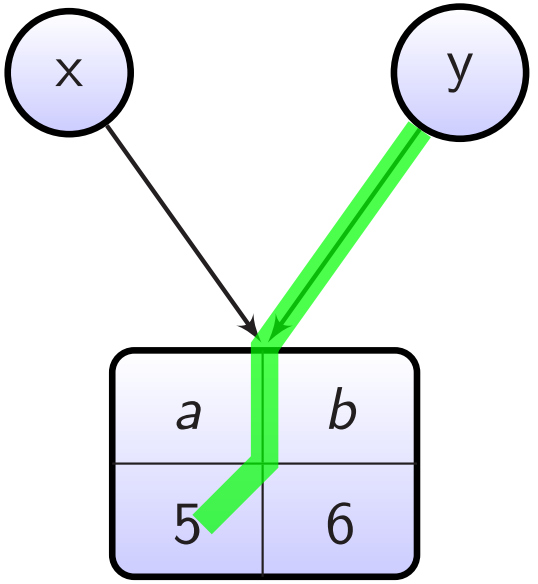
We are looking for a property that holds for all executions of the function.

# Path-Based Interpretation

```
/*c (obj,obj) → any with [ x.b , y.a ] */
function h(x,y) {
  y.a = 1;   // (a)
  y.b = 2;   // (b), violation?
}
```



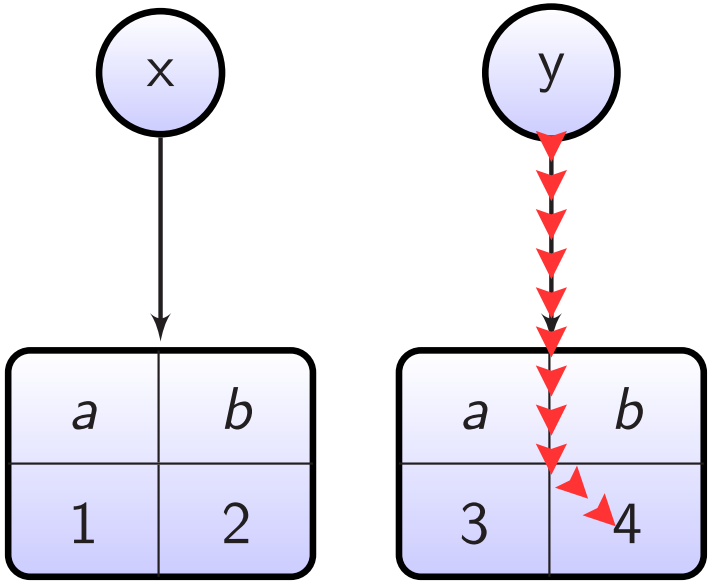

```
h({a:1,b:2},{a:3,b:4});
```



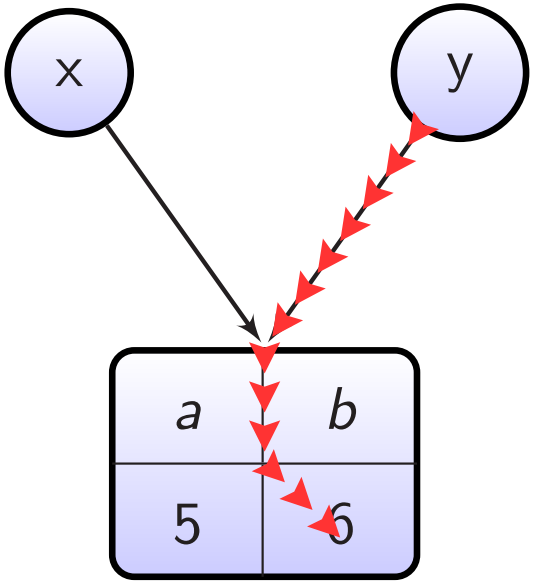

```
var o = {a:5,b:6};
h(o,o);
```

# Path-Based Interpretation

```
/*c (obj,obj) → any with [ x.b , y.a ] */
function h(x,y) {
  y.a = 1;   // (a)
  y.b = 2;   // (b), violation?
}
```



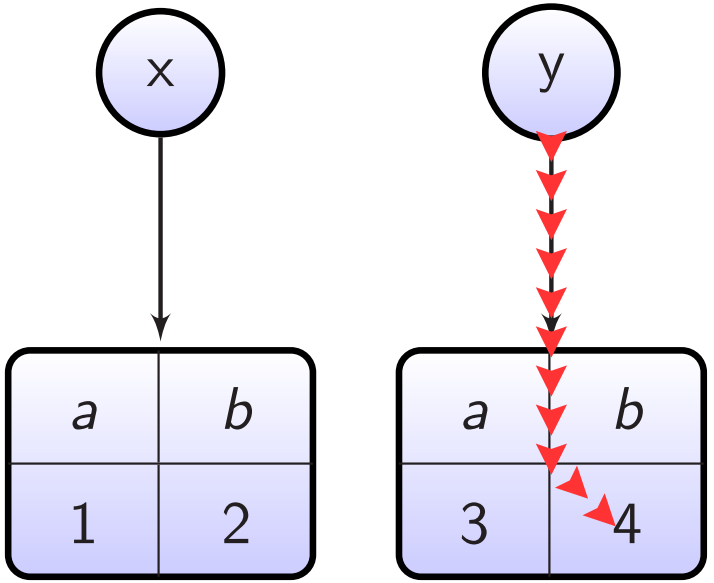

```
h({a:1,b:2},{a:3,b:4});
```



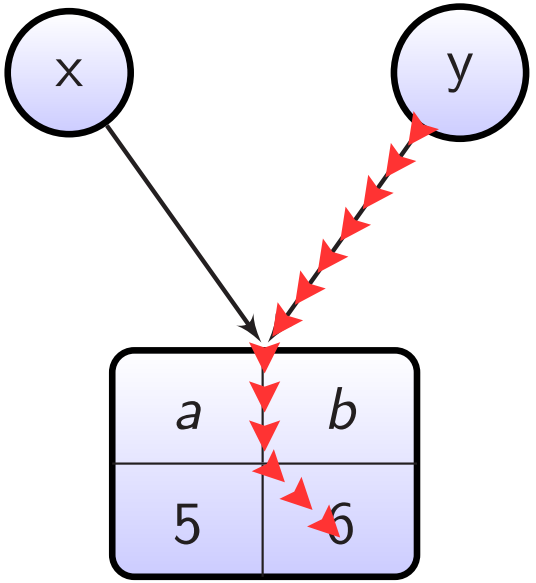

```
var o = {a:5,b:6};
h(o,o);
```

# Path-Based Interpretation

```
/*c (obj,obj) → any with [ x.b , y.a ] */
function h(x,y) {
  y.a = 1;   // (a)
  y.b = 2;   // (b), violation?  YES!
}
```




```
h({a:1,b:2},{a:3,b:4});
```




```
var o = {a:5,b:6};
h(o,o);
```

# Design Principles

UNI FREIBURG

- Pre-State Snapshot
- Dynamic Extent
- Path-Based

# Formalization

## Core Calculus

- lambda calculus with objects (big step semantics)
- expression to install access permissions:
  `permit x :` $L_r, L_w$ `in e`
- marking is done lazy

# Formalization

UNI FREIBURG

## Theorem: Pre-State Snapshot Consistency

The access paths are interpreted with respect to the heap at installation time of the access permission.

## Theorem: Stability of Violation

Introducing new aliases either yields the same access contract violations or yields an inconsistent read after write operation.

# Implementation

UNI FREIBURG

## Prototype Implementation

- offline source-to-source transformation
- library to monitor property reads/writes

# Evaluation

UNI FREIBURG

## Mutation Testing

| Software | types<br>in (%) | types + access perm.<br>in (%) | increase<br>in (%) |
|---|---|---|---|
| Singly-Linked List | 82.0 | 87.3 | 6.4 |
| Richards (V8) | 61.1 | 69.2 | 13.3 |
| Delta-blue (V8) | 75.6 | 87.2 | 15.3 |

# Evaluation

UNI FREIBURG

## Mutation Testing

| Software | types in (%) | types + access perm. in (%) | increase in (%) |
|---|---|---|---|
| Singly-Linked List | 82.0 | 87.3 | 6.4 |
| Richards (V8) | 61.1 | 69.2 | 13.3 |
| Delta-blue (V8) | 75.6 | 87.2 | 15.3 |

## How do we create the access contracts?

We have an inference for them: P. Heidegger, P. Thiemann. A Heuristic Approach for Computing Effects. TOOLS 2011.

# Related Work

UNI FREIBURG


- A. Greenhouse and J. Boyland. An object-oriented effect system. ECOOP 1999.
- J. Smans, B. Jacobs, and F. Piessens. Implicit dynamic frames: Combining dynamic frames and separation logic. ECOOP 2009.
- H. Lehner and P. Müller. Efficient runtime assertion checking of assignable clauses with datagroups. FASE 2010.

# Conclusion

UNI FREIBURG

Access permission contracts

- describe side effects of operations
- are formalized, provably sound and stable
- simplify software development and maintenance
- fix some design flaws of JavaScript

More about JSConTest (implementation, tools, technical report, case studies):
`http://proglang.informatik.uni-freiburg.de/jscontest/`

# Thank You

UNI FREIBURG

Thank you for

your attention!

# More Complicated Example

UNI FREIBURG

```
/*c list.(top) → undf                    ← type signature
     with [this._head, this._length,      ← access
           this._head.next*.next             permission
          ] */
function add(data) {
  var node = {data: data, next: null}, current;
  if (this._head === null) {
    this._head = node;
  } else {
    current = this._head;
    while (current.next) {
      current = current.next;
    }
    current.next = node;
  }
  this._length++;
}
```

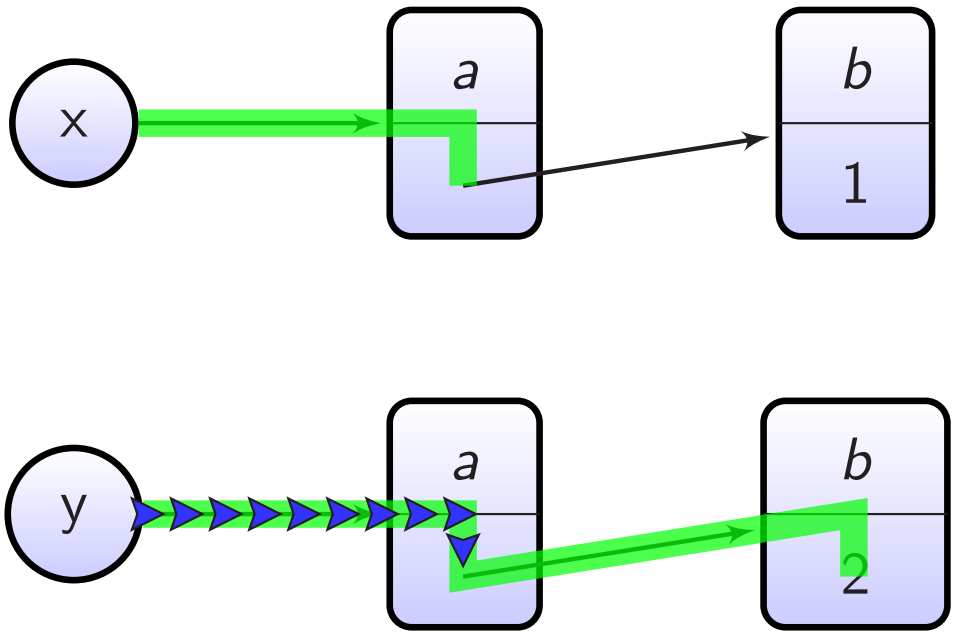
# Example

UNI FREIBURG

```
/*c (obj,obj) → any with [x.b,y.a] */
function h(x,y) {
  y.a = 1;
  y.b = 2;   // violation?
}
```

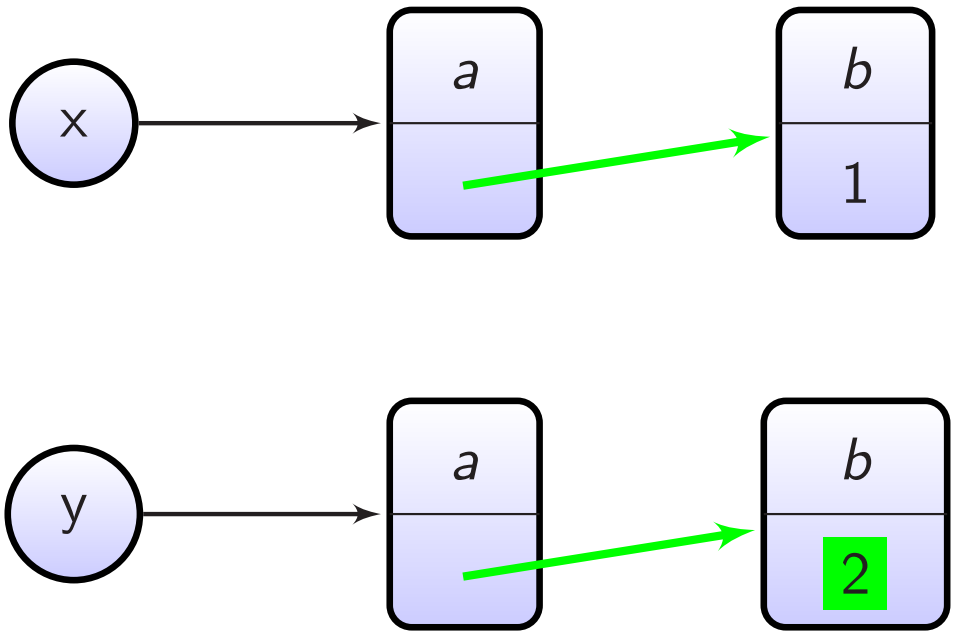
# Path-Based Interpretation – Example

```
/*c (obj,obj) → any with [x.a, y.a, y.a.b] */
function b(x,y) {
  y.a = x.a;
  y.a.b = 42; // violation?
}
```



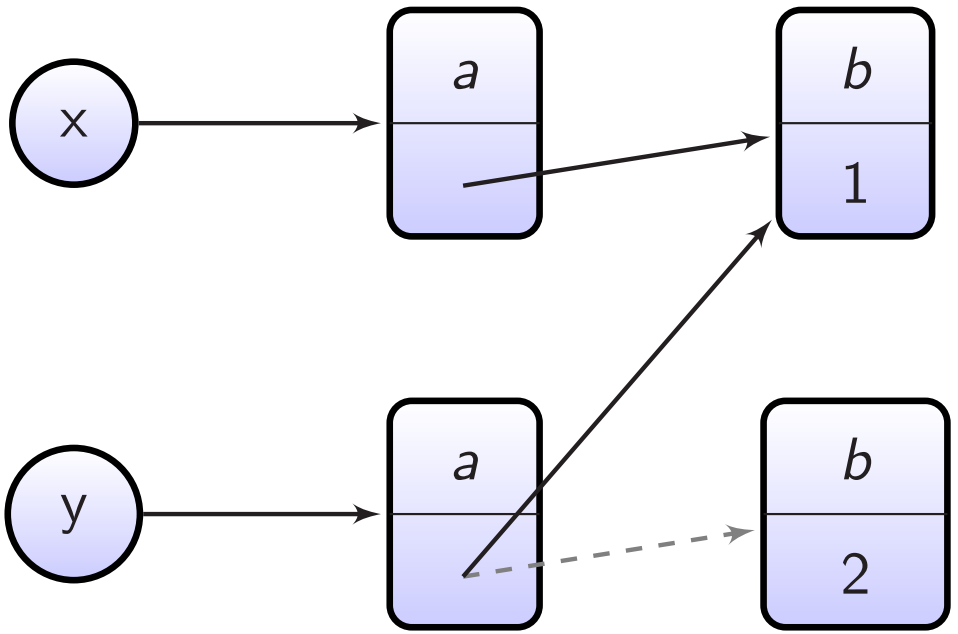
# Path-Based Interpretation – Example

```
/*c (obj,obj) → any with [x.a, y.a, y.a.b] */
function b(x,y) {
  y.a = x.a;
  y.a.b = 42; // violation?
}
```



# Path-Based Interpretation – Example

```
/*c (obj,obj) → any with [x.a, y.a, y.a.b] */
function b(x,y) {
  y.a = x.a;
  y.a.b = 42; // violation?
}
```



# Path-Based Interpretation – Example

```
/*c (obj,obj) → any with [x.a, y.a, y.a.b] */
function b(x,y) {
  y.a = x.a;
  y.a.b = 42; // violation?
}
```