Constraint Semantics for Abstract Read Permissions

John Tang Boyland (UW-Milwaukee/ETH Zurich)
Peter Müller, Malte Schwerhoff, Alexander J. Summers (ETH Zurich)

28th July 2014, FTfJP, Uppsala

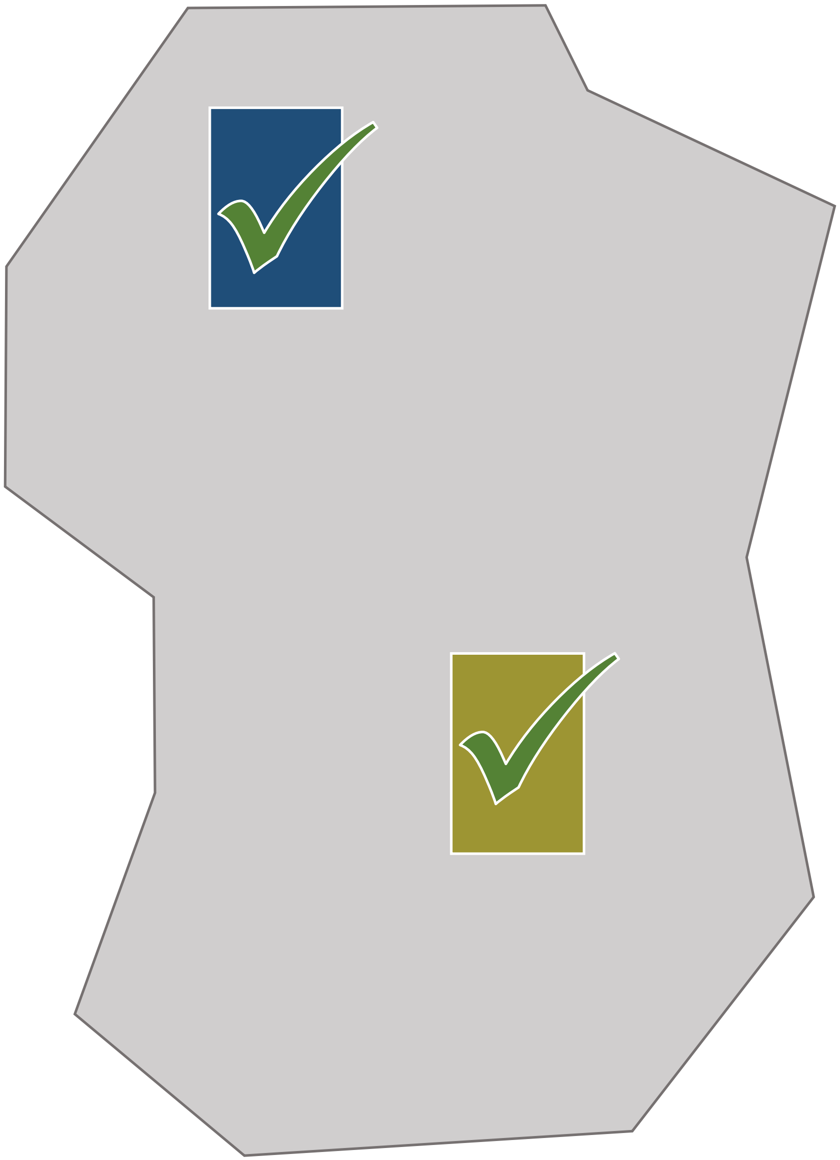
Why Should You Pay Attention?

Unbounded Counting with Fractional Permissions over $\mathbb{Q}$

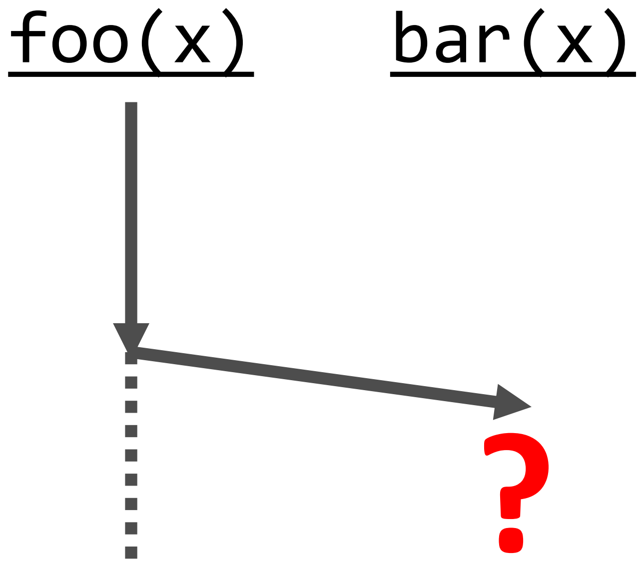
General Framework for Proving Soundness of Permission Assumptions

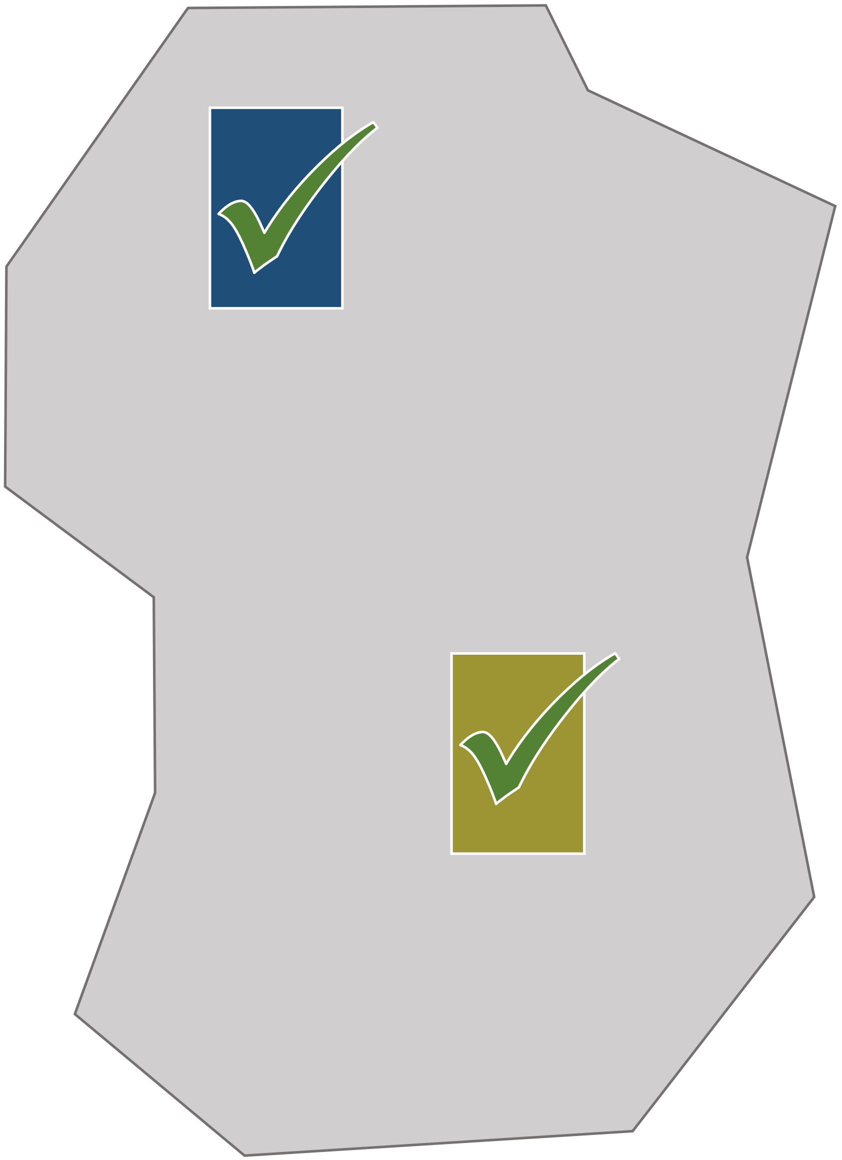
Modular Static Verification + Shared State

`foo(x)`   `bar(x)`

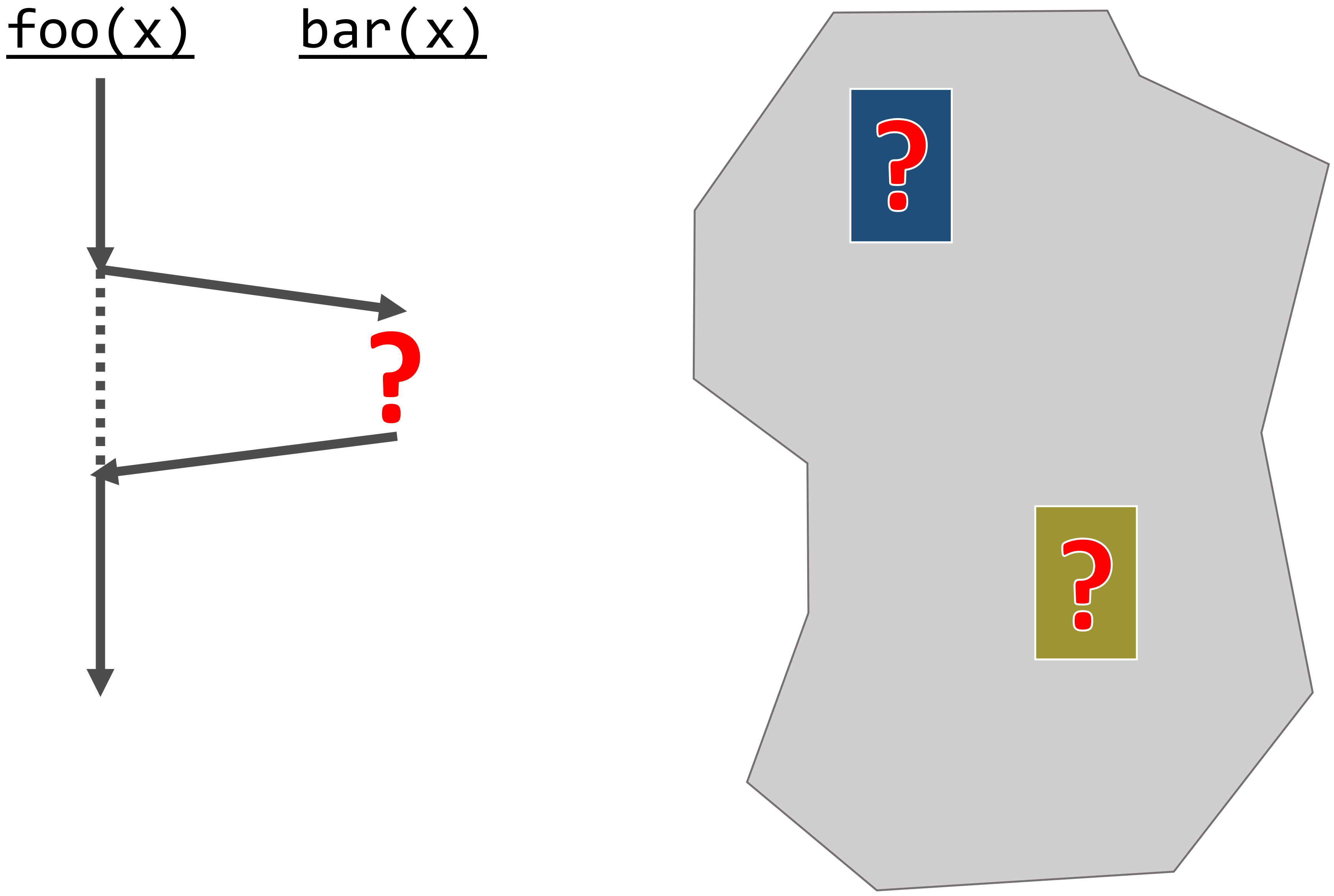


Modular Static Verification + Shared State



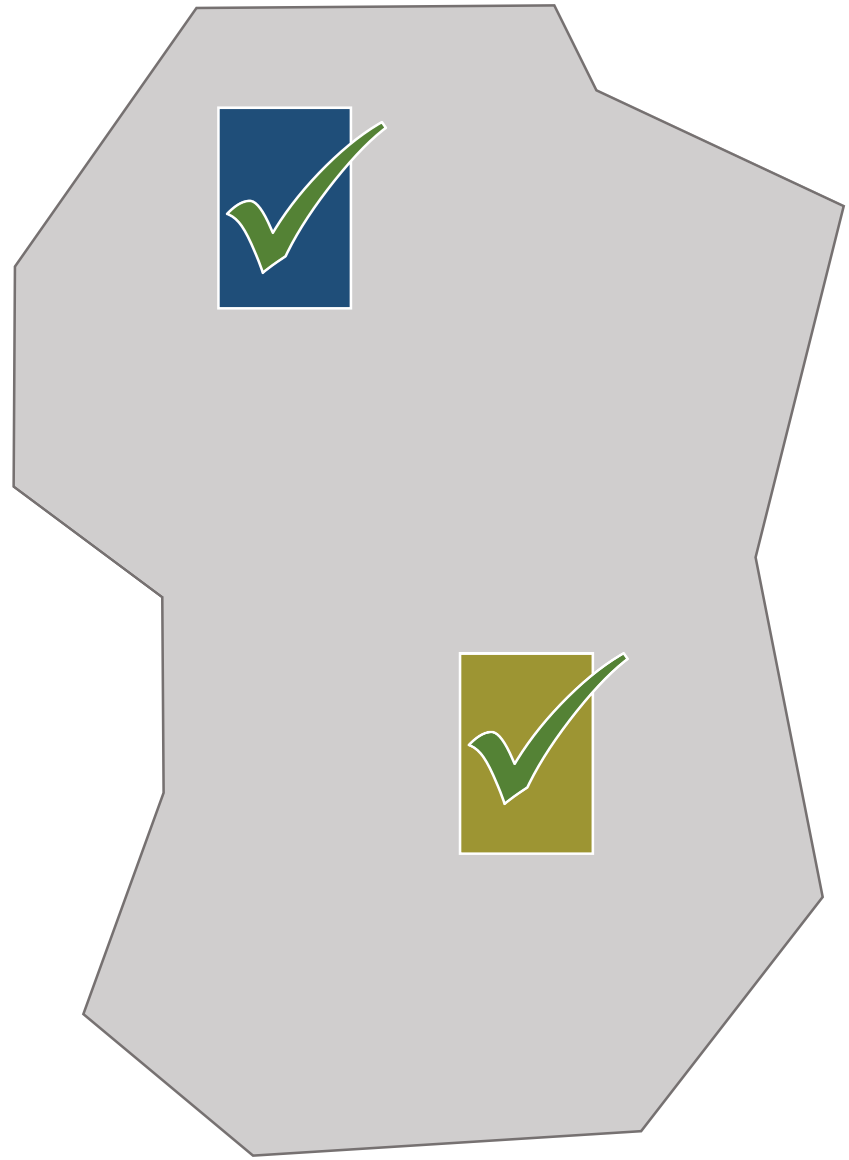


Modular Static Verification + Shared State

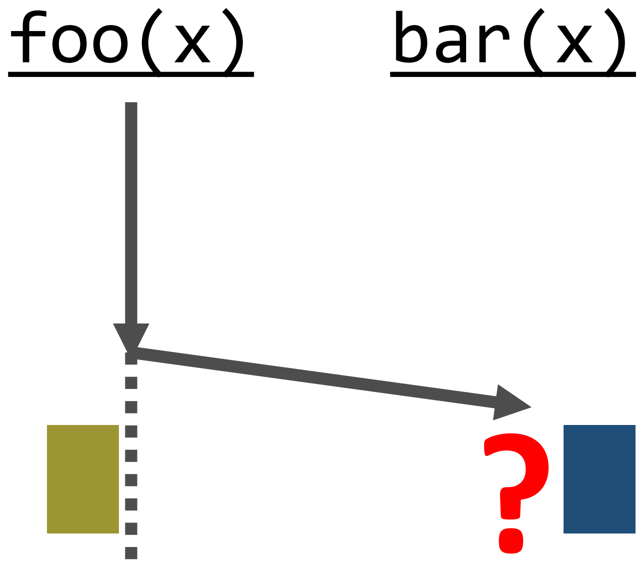


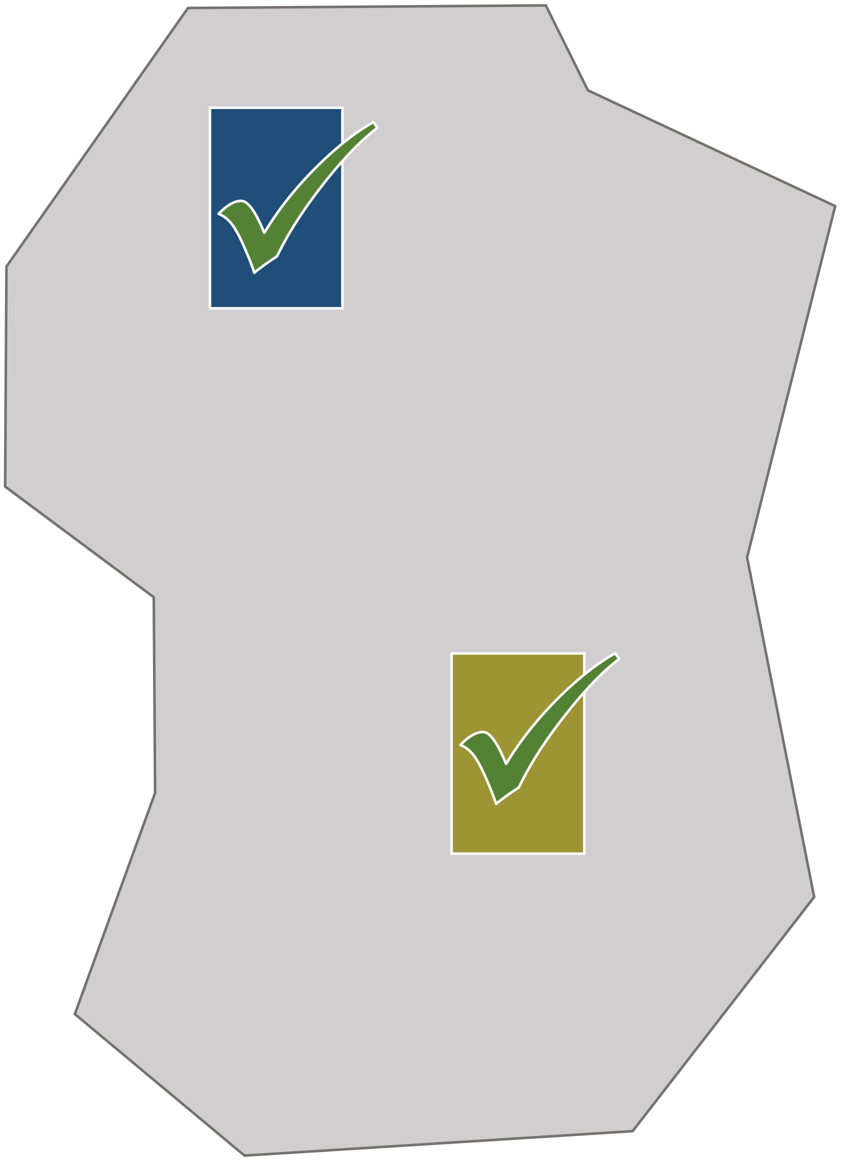
Permissions

`foo(x)` `bar(x)`

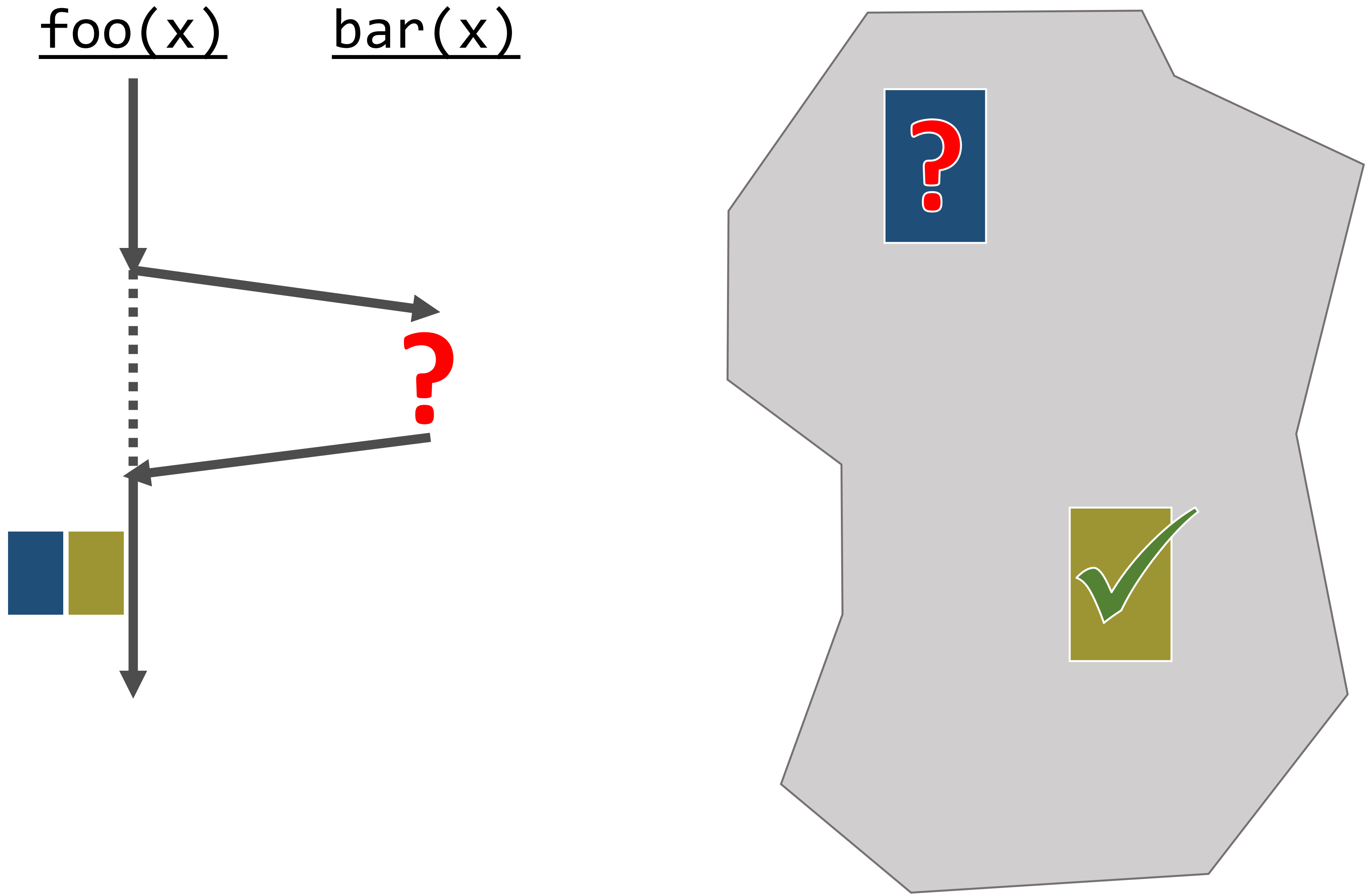


Permission Transfer

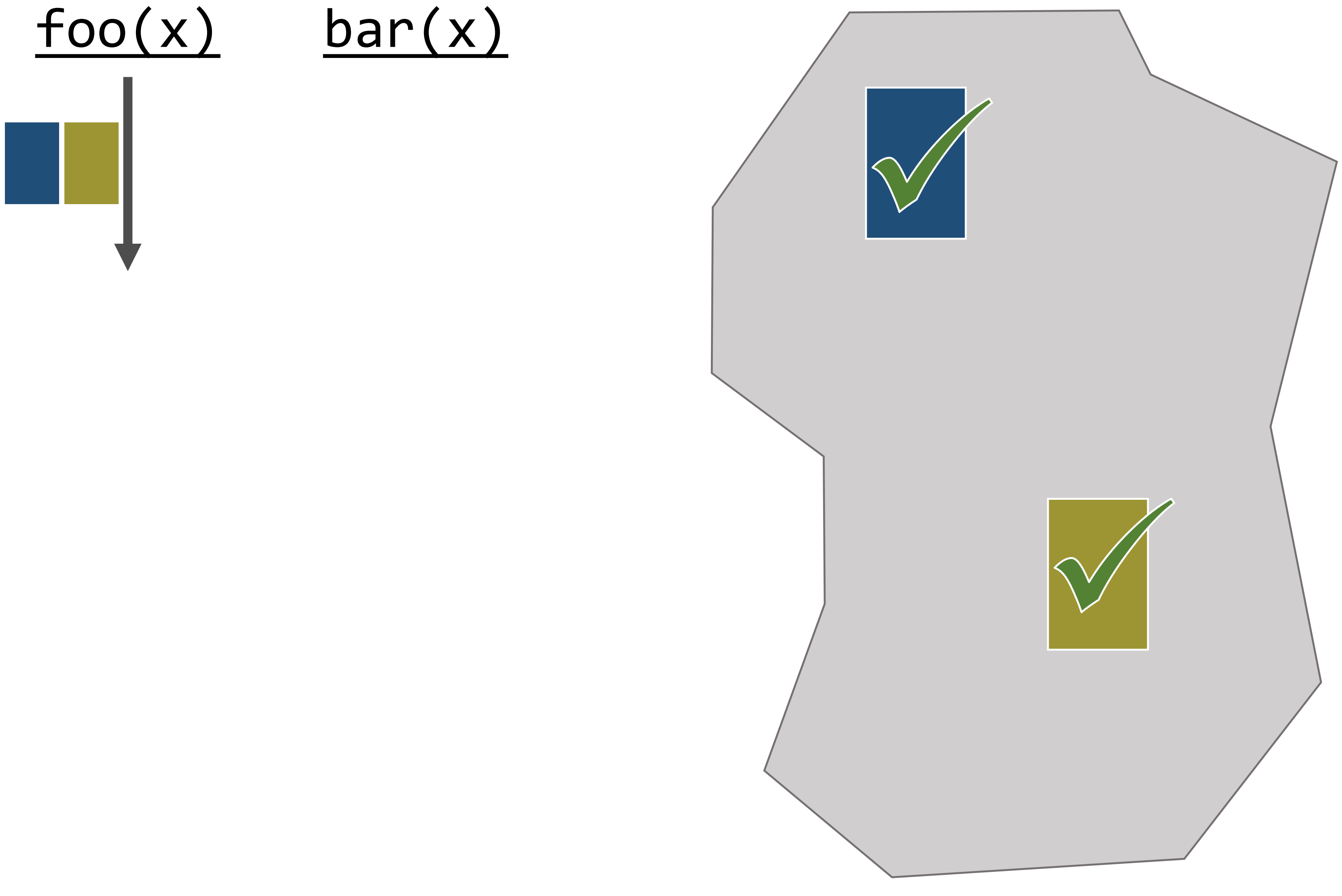




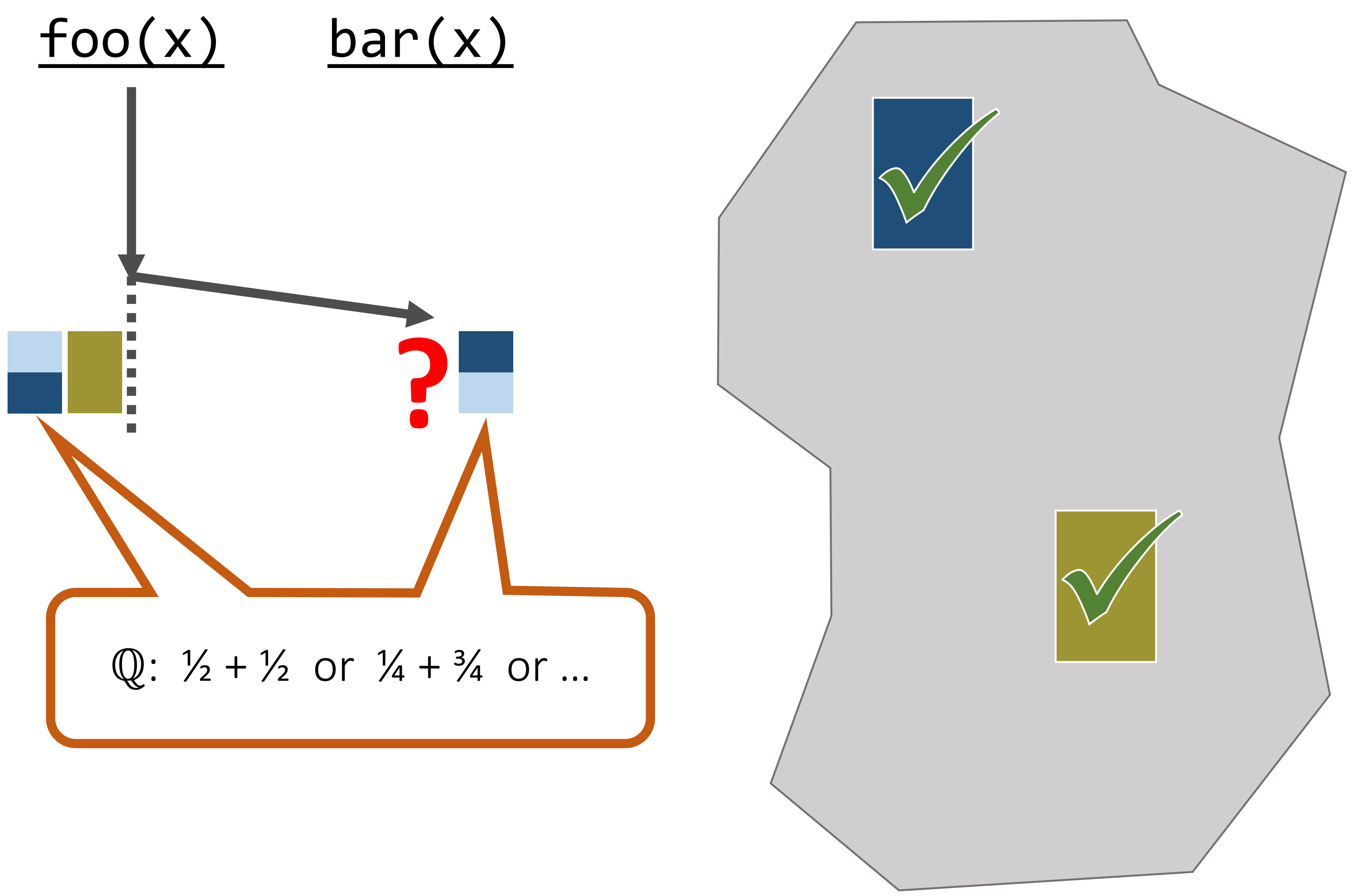
Permission Transfer



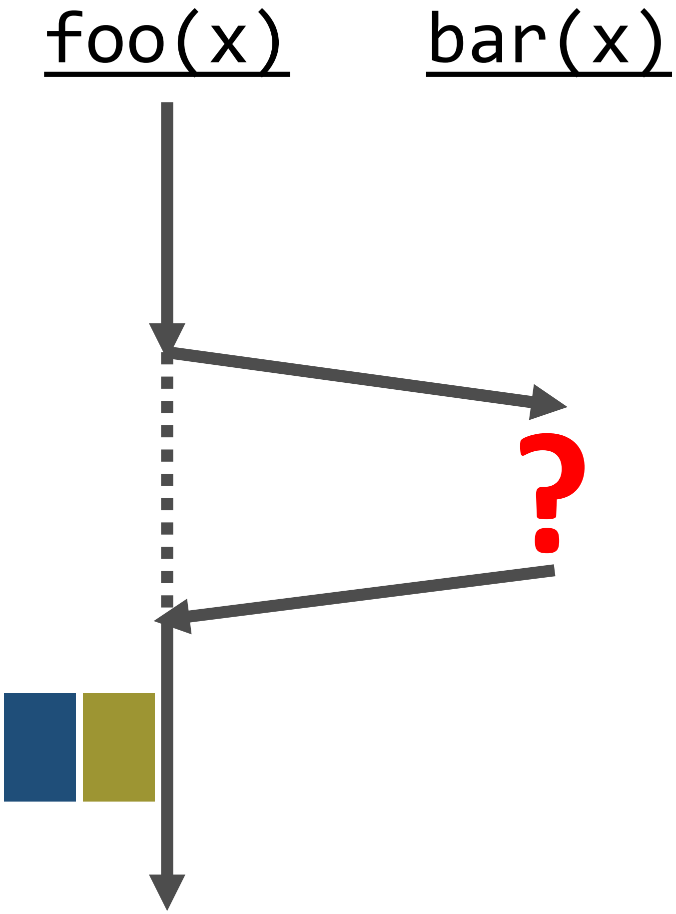
Fractional Permissions

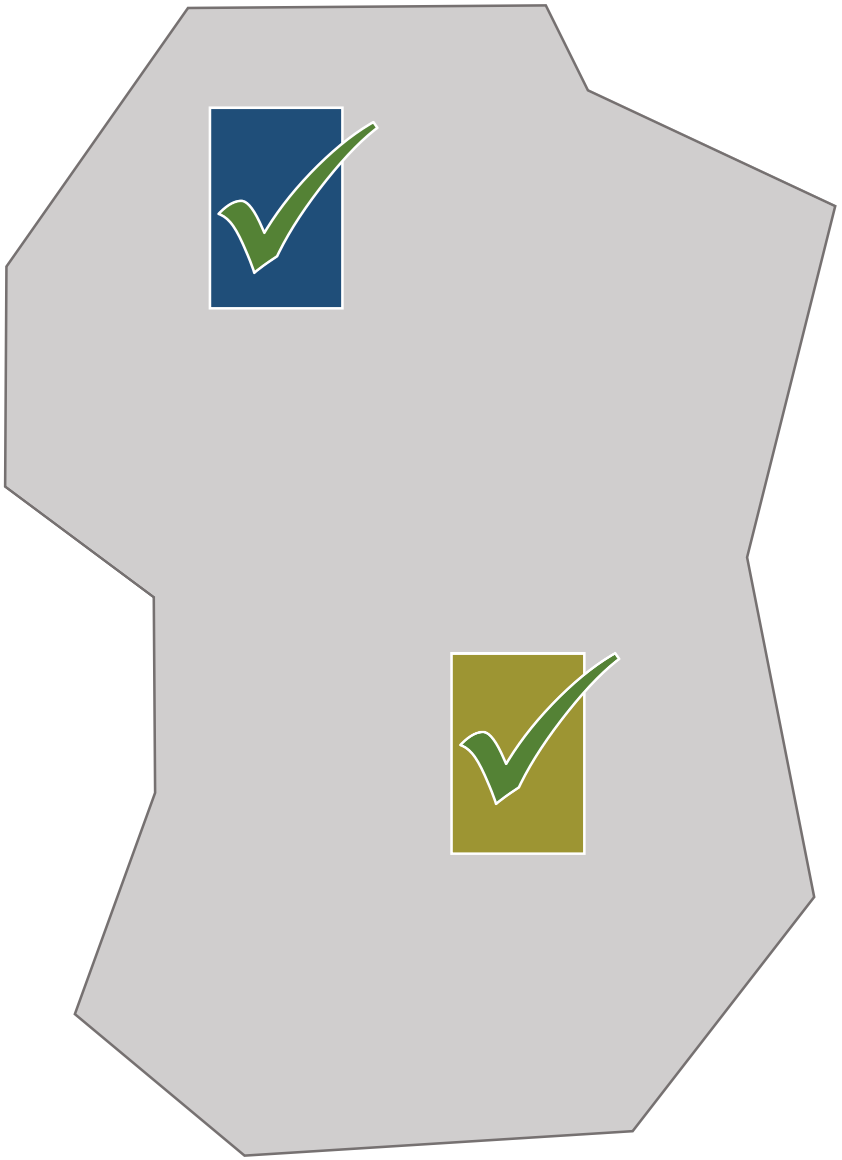


Splitting Fractional Permissions



Merging Fractional Permissions





Iterator Example

```
method visit(it: Iterator, d: Data) {
  var n: Int := 0

  while (it.hasNext()) {
    fork worker(it.next(), d)
    n := n + 1
  }

  ...
}
```

Start with v_1 permissions to `d.f`

```
method worker(e: Element, d: Data) {
  // reads d.f
}
```

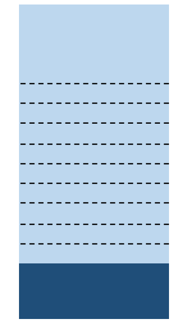
Iterator Example

```
method visit(it: Iterator, d: Data) {
  var n: Int := 0

  while (it.hasNext()) {
    fork worker(it.next(), d)
    n := n + 1
  }

  ...
}
```

Give each worker the **same** v_2 permissions to **`d.f`**



```
method worker(e: Element, d: Data) {
  // reads d.f
}
```

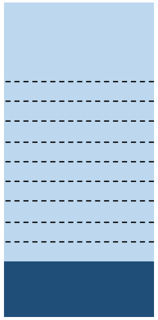
Iterator Example

```
method visit(it: Iterator, d: Data) {
  var n: Int := 0

  while (it.hasNext()) {
    fork worker(it.next(), d)
    n := n + 1
  }

  ...
}
```

Can $(n+1)^{th}$ $\mathbf{v}_2$ still be given away?



```
method worker(e: Element, d: Data) {
  // reads d.f
}
```

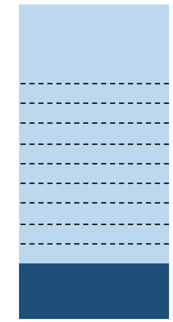
Iterator Example

```
method visit(it: Iterator, d: Data) {
  var n: Int := 0

  while (it.hasNext()) {
    fork worker(it.next(), d)
    n := n + 1
  }

  ...
}
```

Can $(n+1)^{th}$ $\mathbf{v}_2$ still be given away?



```
method worker(e: Element, d: Data) {
  // reads d.f
}
```

Fixing $\mathbf{v}_2$ up-front not possible since $\mathbf{n}$ unbounded ...

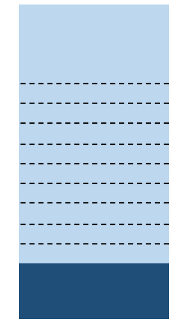
Iterator Example

```
method visit(it: Iterator, d: Data) {
  var n: Int := 0

  while (it.hasNext()) {
    fork worker(it.next(), d)
    n := n + 1
  }

  ...
}
```

Can $(n+1)^{th}$ $\mathbf{v}_2$ still be given away?



```
method worker(e: Element, d: Data) {
  // reads d.f
}
```

... on the other hand, for any $\mathbf{n}$, a suitable $\mathbf{v}_2$ can be chosen

Abstract Read Permissions

(Heule et al., VMCAI'13)

1. Use symbolic values $\mathrm{v_i}$ instead of concrete fractions

2. Constrain $\mathrm{v_i}$ as verification proceeds to make it sufficiently small

Iterator Example with Abstract Read Permissions

```
method visit(it: Iterator, d: Data, v1: Perm) {
  var n: Int := 0
  var v2: Perm := fresh()

  while (it.hasNext()) {

    fork worker(it.next(), d, v2)
    n := n + 1
  }

  ...
}
```

```
method worker(e: Element, d: Data, v: Perm) {
  // reads d.f
}
```

Iterator Example with Abstract Read Permissions

```
method visit(it: Iterator, d: Data, v1: Perm) {
  var n: Int := 0
  var v2: Perm := fresh()

  while (it.hasNext()) {
    assume v2 < v1 - n * v2
    fork worker(it.next(), d, v2)
    n := n + 1
  }

  ...
}
```

Enables $\mathbb{Q}$ to support **unbounded counting** !

```
method worker(e: Element, d: Data, v: Perm) {
  // reads d.f
}
```

Other Permission Model Properties (Short Digression)

```
method visit(it: Iterator, d: Data, v1: Perm) {
  var n: Int := 0
  var v2: Perm := fresh()

  while (it.hasNext()) {
    assume v2 < v1 – n * v2
    fork worker(it.next(), d, v2)
    n := n + 1
  }

  ...
}
```

```
method worker(e: Element, d: Data, v: Perm) {
  // reads d.f
}
```

Recursively fork visit
→ unbounded splitting

Other Permission Model Properties (Short Digression)

```
method visit(it: Iterator, d: Data, v1: Perm) {
  var n: Int := 0
  var v2: Perm := fresh()

  while (it.hasNext()) {
    assume v2 < v1 - n * v2
    fork worker(it.next(), d, v2)
    n := n + 1
  }

  ...
}
```

```
method worker(e: Element, d: Data, v: Perm) {
  // reads d.f
}
```

Abstract predicates
→ permission multiplication

Problem

Permission Assumptions Sound?

$$\Leftrightarrow$$

Constraint System Satisfiable?

Abstract Read Permissions (Heule et al., VMCAI'13)

- Must avoid unsound assumptions (in a modular way)
  - $v < v$
  - $v_1 < v_2 \; \&\& \; v_2 < v_1$

- Factoring key property for constraint satisfiability out of general soundness proof is not straight-forward

- Heule et al. therefore allow only limited application of abstract read permissions
  - Ensures satisfiable constraints
  - **Can't support unbounded counting**

Key Property

Layered Constraints

Layered Constraints

¼, ½, ¾, … $q \in \mathbb{Q}^+$

v_1 v_3

v_4 v_7

v_6

v_5 v_2 v_8

0 0

Layered Constraints

Partially ordered set of variables $(V, \prec)$

$v_2 \prec v_1 \triangleq$ "v_2 layered below v_1"

☑

Extend $\prec$ to $v \prec E$

$E ::= q \mid v \mid E + E \mid E * E \mid E - E \mid E / E$

☐

Define *layered* for sets C of permission inequalities $v < E$

☐

Extending $\prec$ to E

$$\frac{q \in \mathbb{Q}^{+}}{v \prec q} \qquad \frac{v \prec E_1 \quad v \prec E_2}{v \prec E_1 \odot E_2} \qquad \frac{v \prec v' \quad v' \prec E}{v \prec E - v'}$$

with $\odot \in \{+, *, /\}$

Extending $\prec$ to E

$$\frac{q \in \mathbb{Q}^+}{v \prec q} \qquad \frac{v \prec E_1 \quad v \prec E_2}{v \prec E_1 \odot E_2} \qquad \frac{v \prec v' \quad v' \prec E}{v \prec E - v'}$$

with $\odot \in \{+, *, /\}$

Ensures that $v' < E$ *could* be added to $\mathbf{C}$

Extending $\prec$ to E

$$\frac{q \in \mathbb{Q}^+}{v \prec q}$$

$$\frac{v \prec E_1 \qquad v \prec E_2}{v \prec E_1 \odot E_2}$$

with $\odot \in \{+, *, /\}$

$$\frac{v \prec v' \qquad v' \prec E}{v \prec E - v'}$$

Ensures that v is layered below v'

Layered and Layerable Constraints

- A set $\mathbf{C}$ of permission inequalities $\mathbf{v} < \mathbf{E}$ is layered w.r.t. to $\prec$ if

$$\forall\ (\mathrm{v} < \mathrm{E}) \in \mathrm{C}\ \cdot \mathrm{v} \prec \mathrm{E}$$

Layered and Layerable Constraints

- A set C of permission inequalities $\mathrm{v} < \mathrm{E}$ is layered w.r.t. to $\prec$ if

$$\forall\, (\mathrm{v} < \mathrm{E}) \in \mathrm{C} \;\cdot$$
$$\exists\, (\mathrm{v'} < \mathrm{E'}) \cdot$$
$$(\mathrm{v'} < \mathrm{E'} \Rightarrow \mathrm{v} < \mathrm{E}) \;\;\wedge\;\; \mathrm{v'} \prec \mathrm{E'}$$

Makes simple derivation system more expressive

- A set C is *layerable* if there exists a $\prec$ s.t. C is layered w.r.t. to $\prec$

Theorem: If a constraint system is layerable, then it is satisfiable

Iterator Example Revisited

$$\frac{q \in \mathbb{Q}^+}{v \prec q} \qquad \frac{v \prec E_1 \quad v \prec E_2}{v \prec E_1 \odot E_2} \qquad \frac{v \prec v' \quad v' \prec E}{v \prec E - v'}$$

with $\odot \in \{+, *, /\}$

Constraint from iterator example

$$v_2 < v_1 - n * v_2$$

is layered if

$$v_2 \prec v_1 - n * v_2$$

is derivable

Iterator Example Revisited

$$\frac{q \in \mathbb{Q}^+}{v \prec q} \qquad \frac{v \prec E_1 \quad v \prec E_2}{v \prec E_1 \odot E_2} \qquad \frac{v \prec v' \quad v' \prec E}{v \prec E - v'}$$

with $\odot \in \{+, *, /\}$

Constraint from iterator example

$$v_2 < v_1 - n * v_2$$

is layered if

$$v_2 \prec v_1 - n * v_2 \iff v_2 < v_1 / (n + 1)$$

is derivable

Iterator Example Revisited

$$\frac{q \in \mathbb{Q}^+}{v \prec q}$$

$$\frac{v \prec E_1 \qquad v \prec E_2}{v \prec E_1 \circledcirc E_2}$$

with $\circledcirc \in \{+, *, /\}$

$$\frac{v \prec v' \qquad v' \prec E}{v \prec E - v'}$$

$$\frac{v_2 \prec v_1 \qquad \dfrac{(n + 1) \in \mathbb{Q}^+}{v_2 \prec (n + 1)}}{v_2 \prec v_1 \,/\, (n + 1)}$$

Using our Work

1. Define a suitable $\prec$-relation, e.g., *introduced after*

```
method visit(v1: Perm) {
  ...
  var v2: Perm := fresh()
  ...
 }
```

2. Define a methodology for generating layered constraints w.r.t. to chosen $\prec$ (soundness proof)

3. Turn constraints into free assumptions

Related Work

- Counting Permissions (Bornat et al., POPL'05)
  - Neither divisibility nor multiplication

- Compound Models (Dockins et al., APLAS'09, Leino et al., ESOP'09)
  - No multiplication
  - Potentially slow due to disjunctions

- $Z[\varepsilon]^+$ (Boyland, LNCS Volume 7850, 2013)
  - Satisfies all three properties
  - Complex and subtle model
  - No existing implementation (as far as we know)

Conclusion

- Identified a property that guarantees satisfiability of constraints over fractional permissions over $\mathbb{Q}$

- Formalised a derivation system enforcing the property

- Enabled fractional permissions over $\mathbb{Q}$ to support unbounded counting

- Factor soundness proof for permission book-keeping out of general soundness proof

- Future work: define methodology for fully exploiting layerable constraints in a front-end tool

Questions?

www.pm.inf.ethz.ch
malte.schwerhoff@inf.ethz.ch