

# Stabilize

## and other integrity traits

Mark S. Miller



Chip Morningstar



Richard Gibson



Mathieu Hofman



105th Plenary  
December 2024

TC  
39

# Stabilize

## and other integrity traits

Mark S. Miller



Chip Morningstar



Richard Gibson



Mathieu Hofman



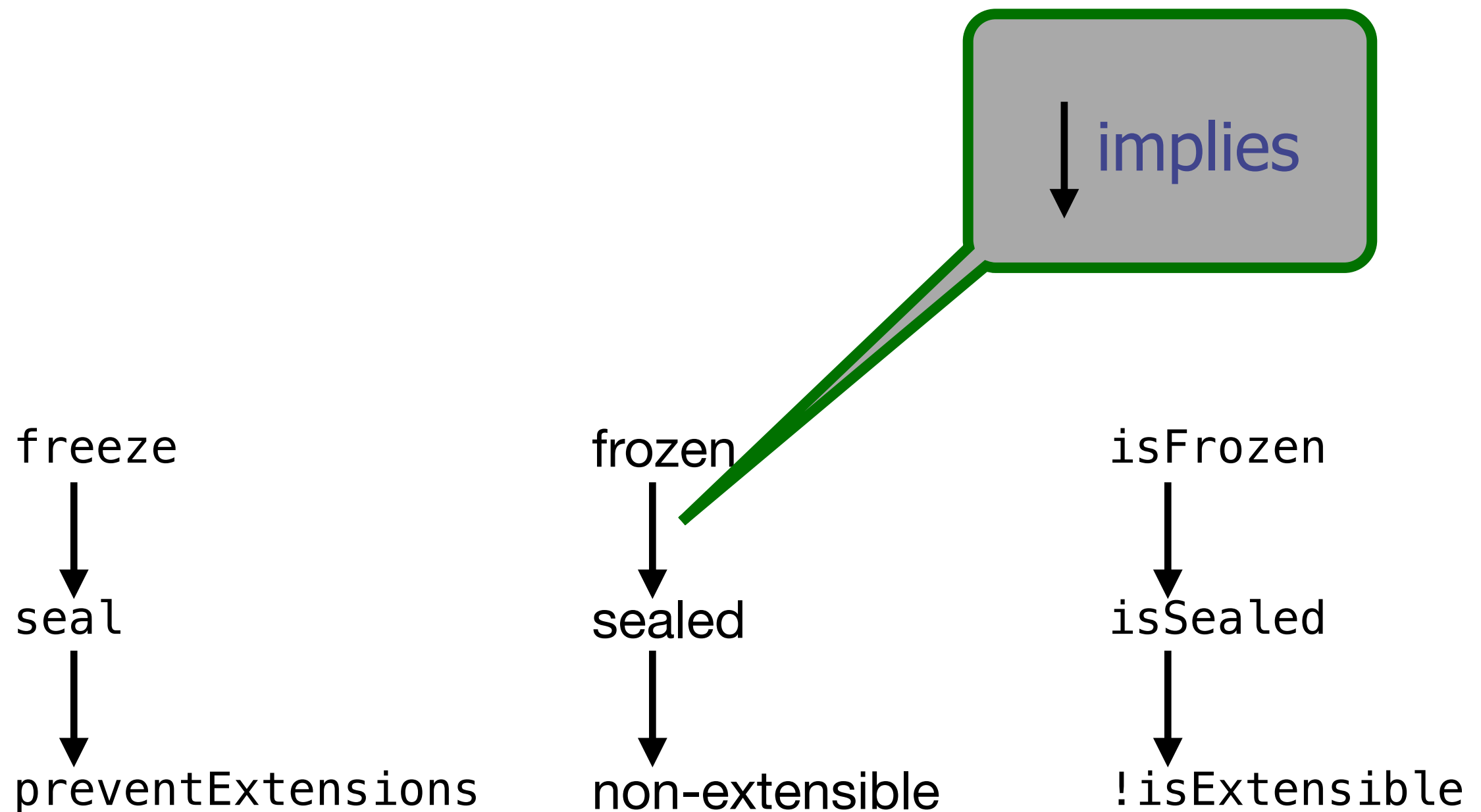
JS Structs Working Group  
November 2024

105th Plenary  
December 2024



# Current Integrity “Levels”

---



# Current Integrity “Levels”

---

- Monotonic one-way switch
- Stronger object invariants
- Proxy is X iff target is X

freeze  
↓  
seal  
↓  
preventExtensions

frozen  
↓  
sealed  
↓  
non-extensible

isFrozen  
↓  
isSealed  
↓  
!isExtensible

# Current Integrity “Levels”

---

- **Explicit** vs Emergent
- 2 Proxy traps per **explicit** integrity “level”

freeze  
↓  
seal  
↓  
preventExtensions

frozen  
↓  
sealed  
↓  
**non-extensible**

isFrozen  
↓  
isSealed  
↓  
!isExtensible

# Need better integrity support

---

frozen



sealed



**non-extensible**

Today:  
harden is deep freeze

Hardened JS:  
hardens all primordials

3 weaknesses

# First Try

---

*stable*



frozen



sealed



*non-extensible*

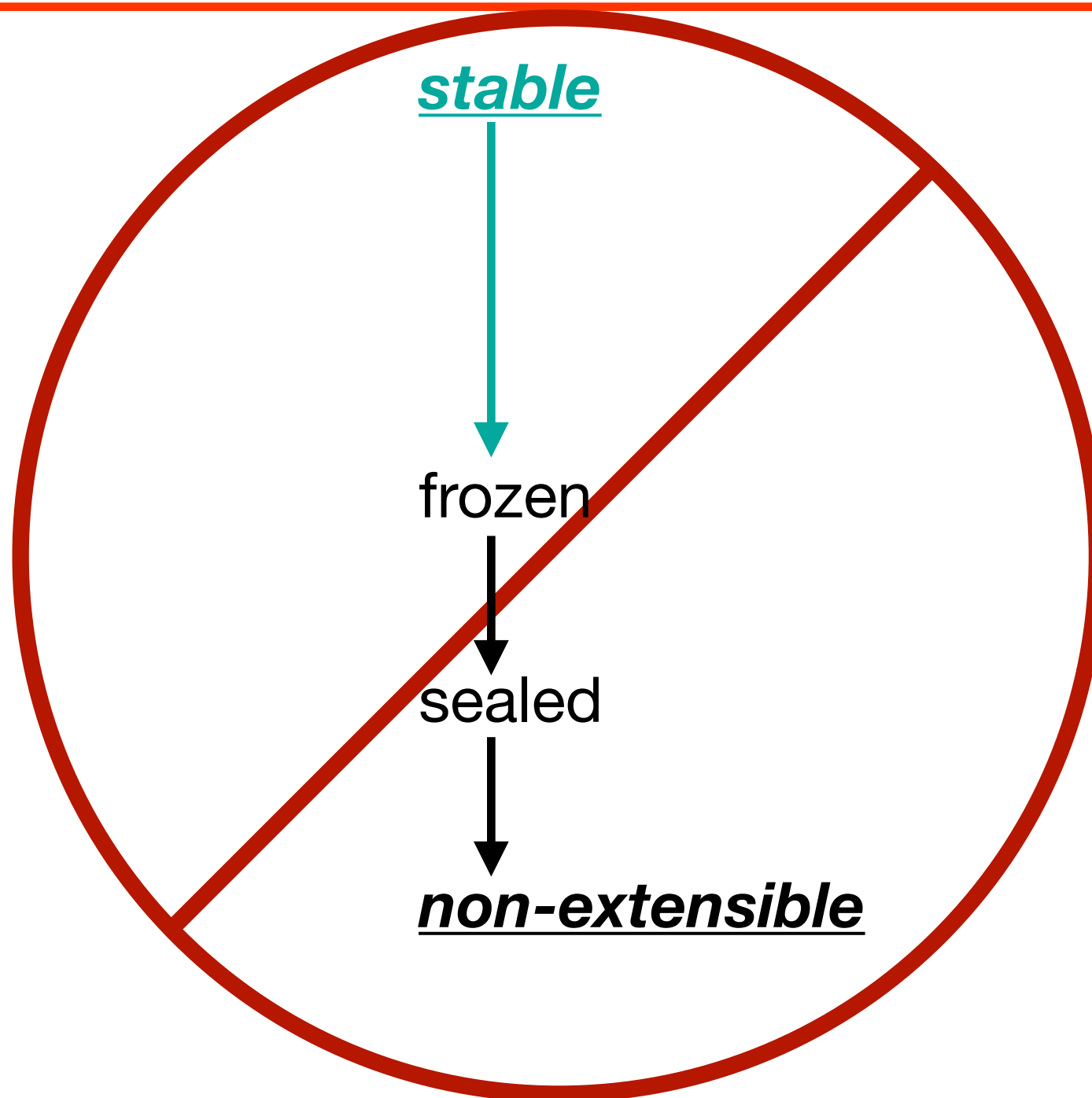
Want:  
harden is deep stabilize

Hardened JS:  
hardens all primordials

Strong enough

# First Try. But Structs Cannot be Frozen

---





# First Try. But Structs Cannot be Frozen

---

stable



frozen



sealed



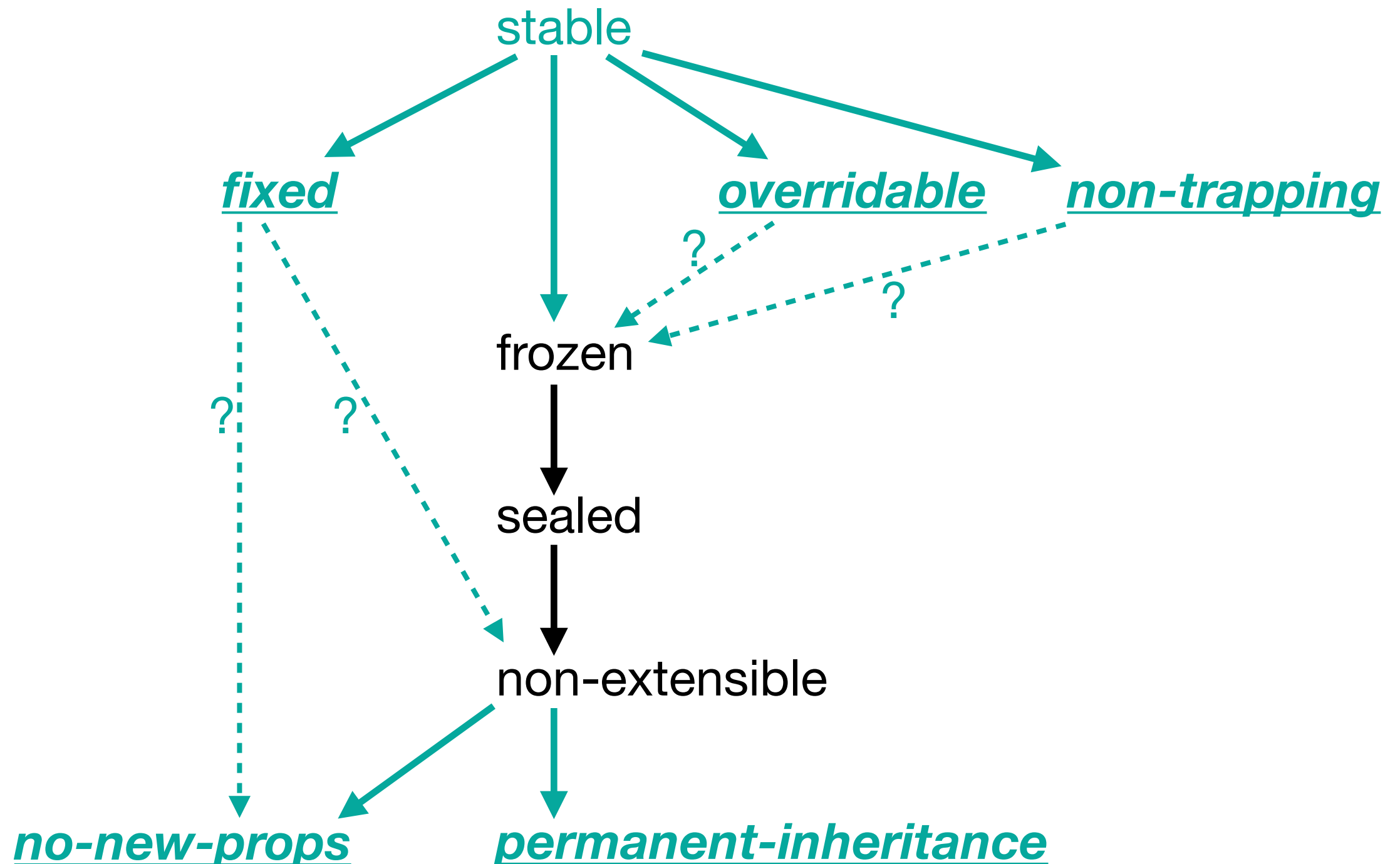
non-extensible

*“Only two ways to [...],  
one is to bundle,  
the other is to unbundle.”*

—Jim Barksdale

# Maximally Unbundled Integrity “Traits”

---



*fixed*

# Mitigate return-override mistake

---

No *stamping*

Fixed shape structs

Retcon windowProxy

Virtualizable weakness

*fixed*

```
class Superclass {  
  constructor(key) { return key; }  
}
```

```
class Subclass extends Superclass {  
  #value  
  constructor(key, value) {  
    super(key);  
    this.#value = value;  
  }  
}
```

```
new Subclass(struct, 'a');
```

*no-new-props*

*perm*

# overridable

## Mitigate assignment-override mistake

---

Fix biggest integrity *deterrent*  
Allow override by assignment  
(better globally if possible)

```
Object.freeze(Object.prototype);

function Point(x, y) {
  this.x = y; this.y = y;
}

Point.prototype.toString =
function () {
  return `<${this.x},${this.y}`;
};
```

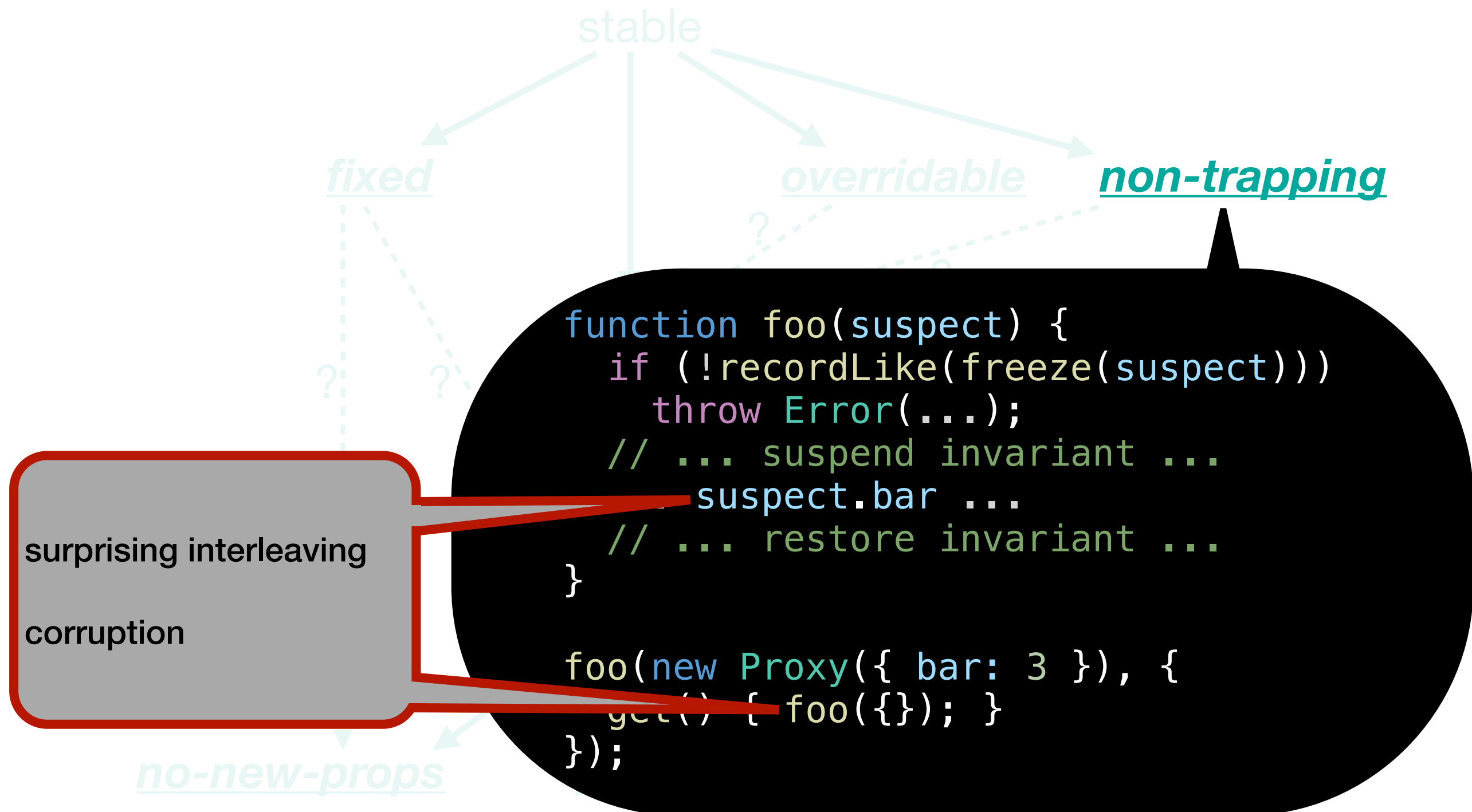
no-new-props

per

# *non-trapping*

## Mitigate proxy reentrancy hazards

---



# *non-trapping*

## Mitigate proxy reentrancy hazards

records were just pojos

No handler traps  
(or equivalent for exotics)

pojos help defensiveness

stable

overridable

*non-trapping*

```
function foo(suspect) {  
  if (!recordLike(stabilize(suspect)))  
    throw Error(...);  
  // ... suspend invariant ...  
  ... suspect.bar ...  
  // ... restore invariant ...  
}
```

```
foo(new Proxy({ bar: 3 }), {  
  get() { foo({}); };  
});
```

*no-new-props*

# *permanent-inheritance*

Retcon windowProxy, Object.prototype

---

Too much magic

Make objects less exotic

*fixed*

*able*

*non-trapping*

```
> Object.setPrototypeOf(window, {});  
Uncaught TypeError: Immutable prototype object  
'#<Window>' cannot have their prototype set  
  
> Object.setPrototypeOf(Object.prototype, {});  
Uncaught TypeError: Immutable prototype object  
'Object.prototype' cannot have their prototype set
```

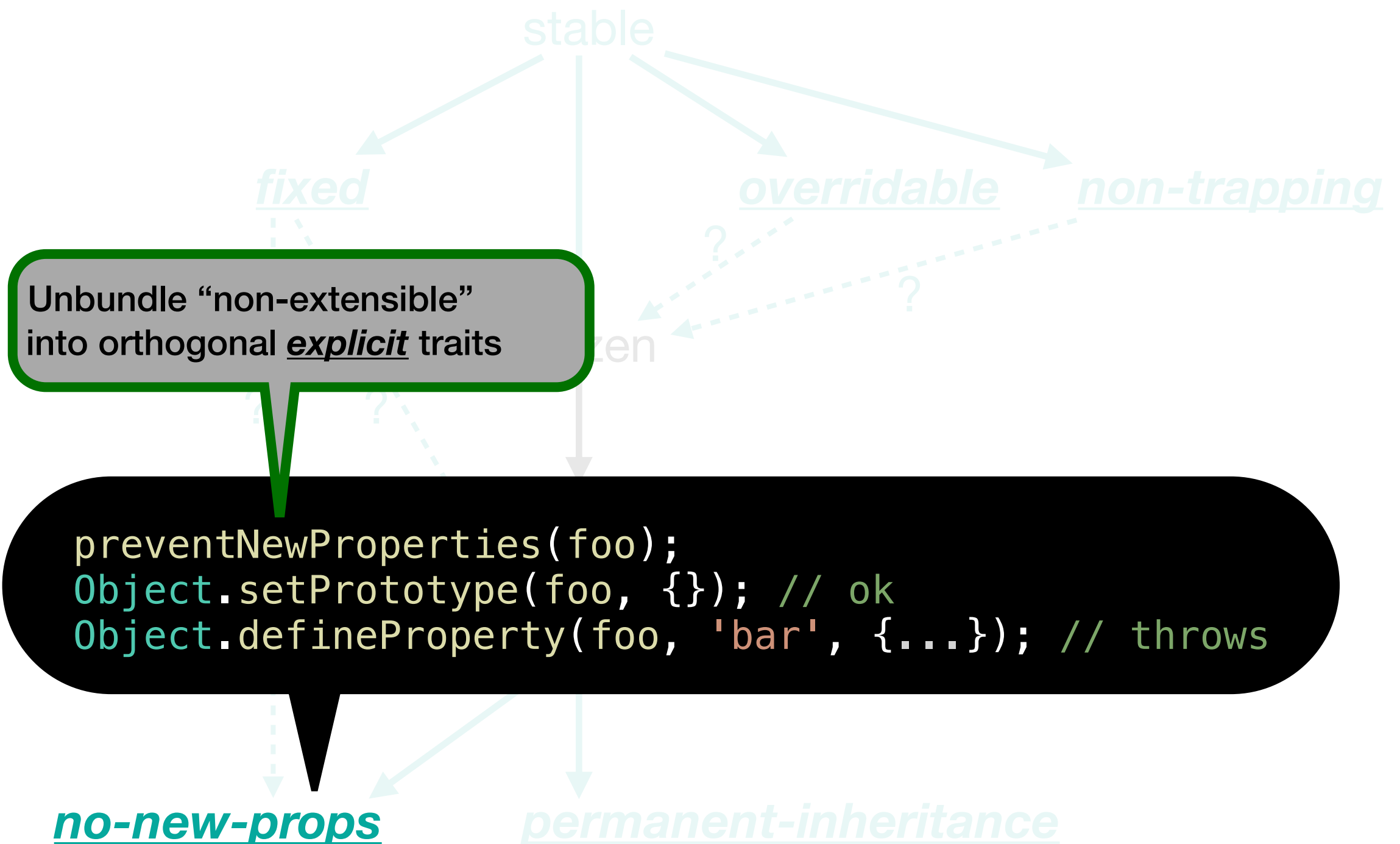
*no-new-props*

*permanent-inheritance*

# *no-new-props*

## Rest of non-extensible unbundling

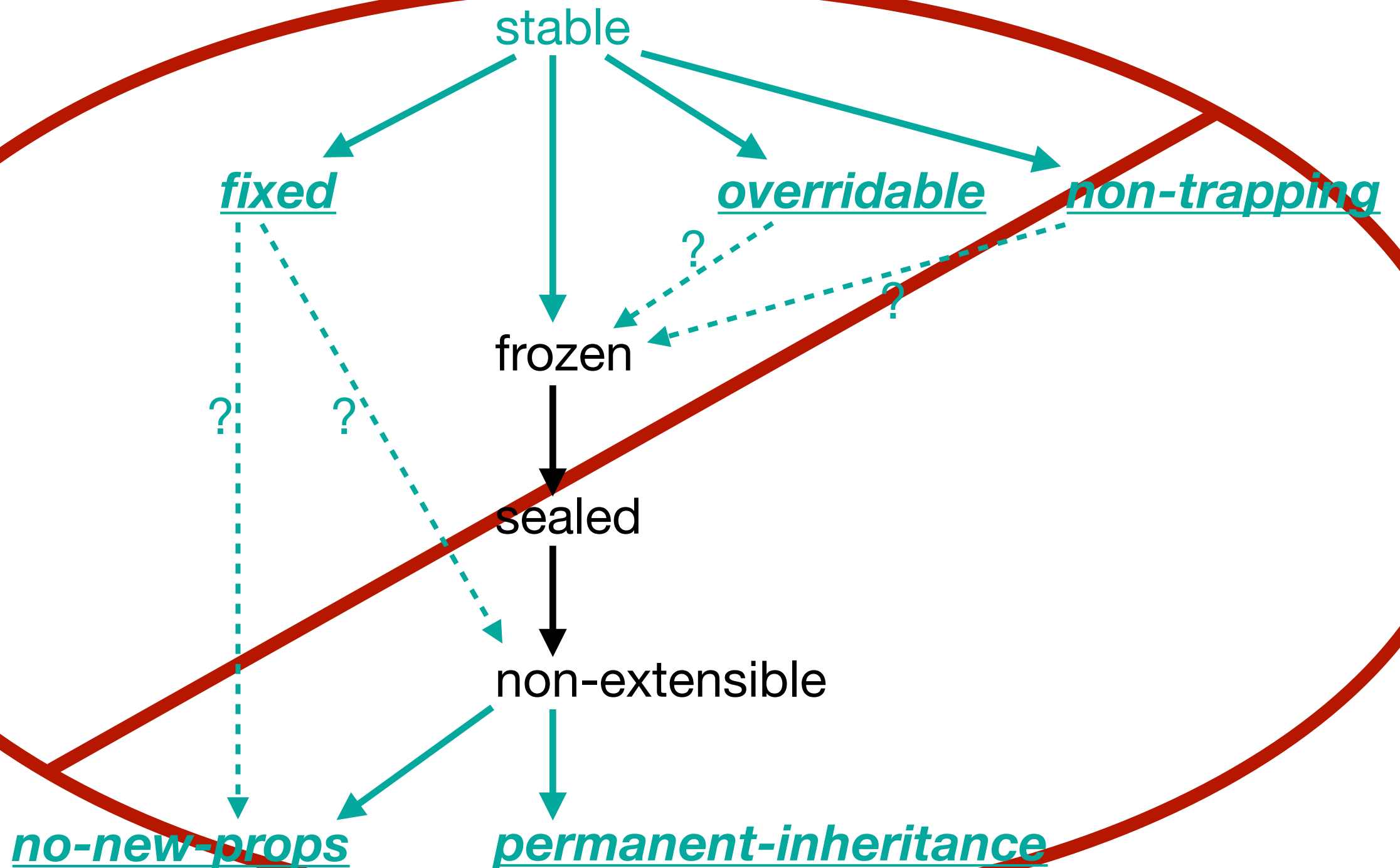
---



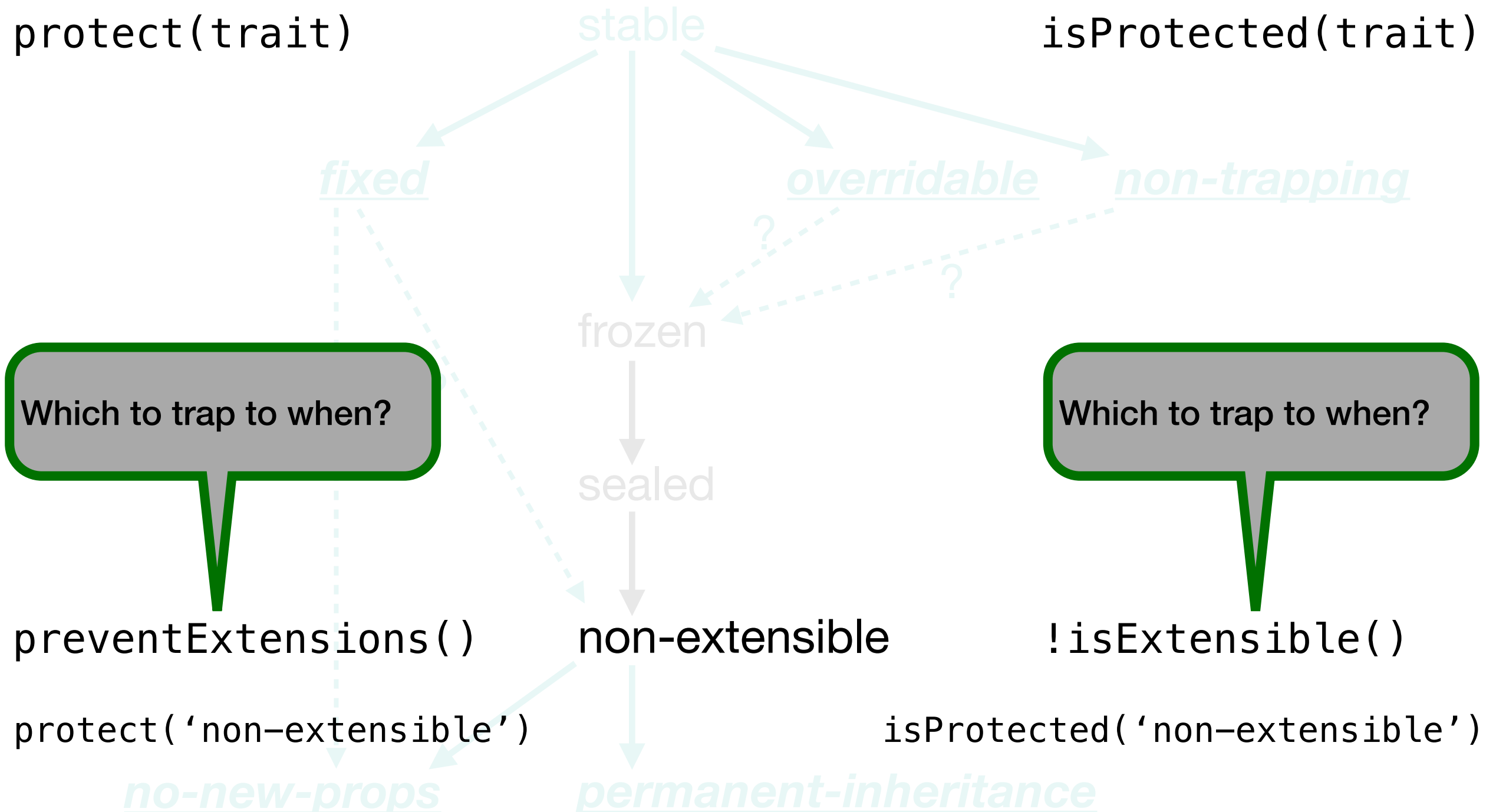


# 10 New Proxy Traps — Too many!

---

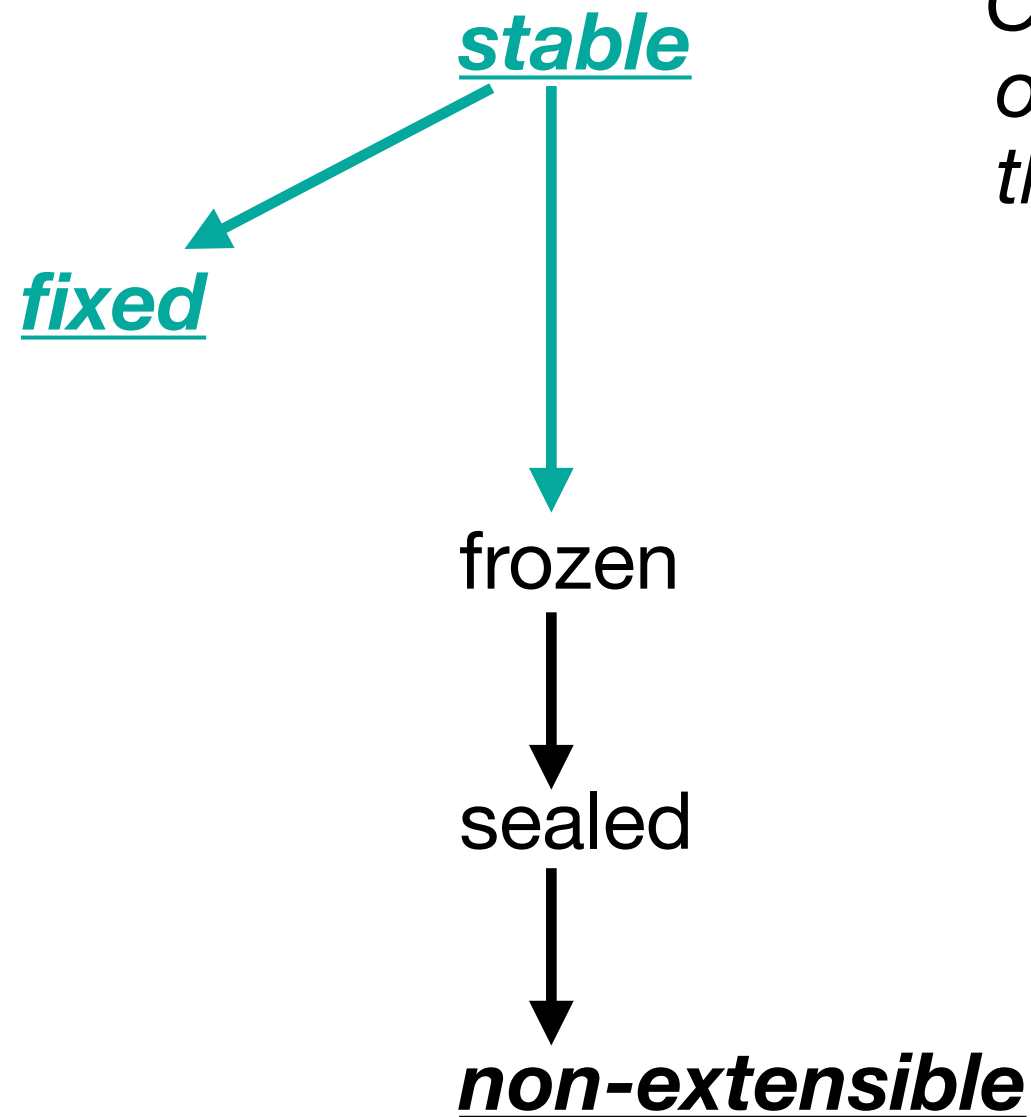


## 2 New Parameterized Proxy Traps



# Minimal Proposed Integrity “Traits”

---



*“Only two ways to [...],  
one is to bundle,  
the other is to unbundle.”*

—Jim Barksdale

# Questions?

---

