Stabilize status update

Hopes and Dreams

Mark S. Miller — Agoric

Chip Morningstar — METAMASK

Richard Gibson — Agoric

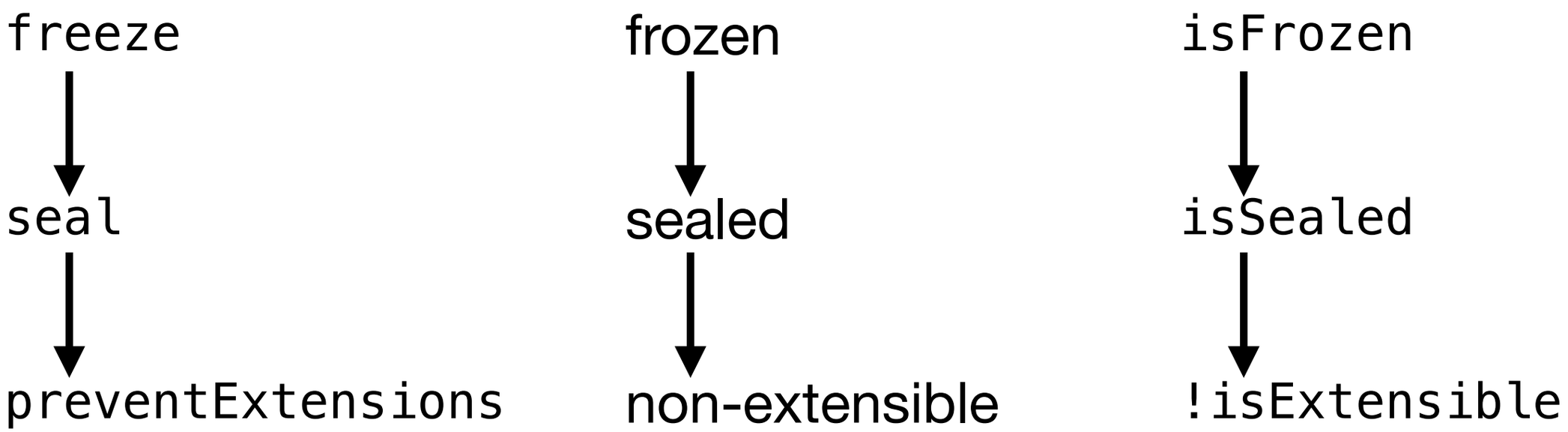
Mathieu Hofman — Agoric

105th Plenary

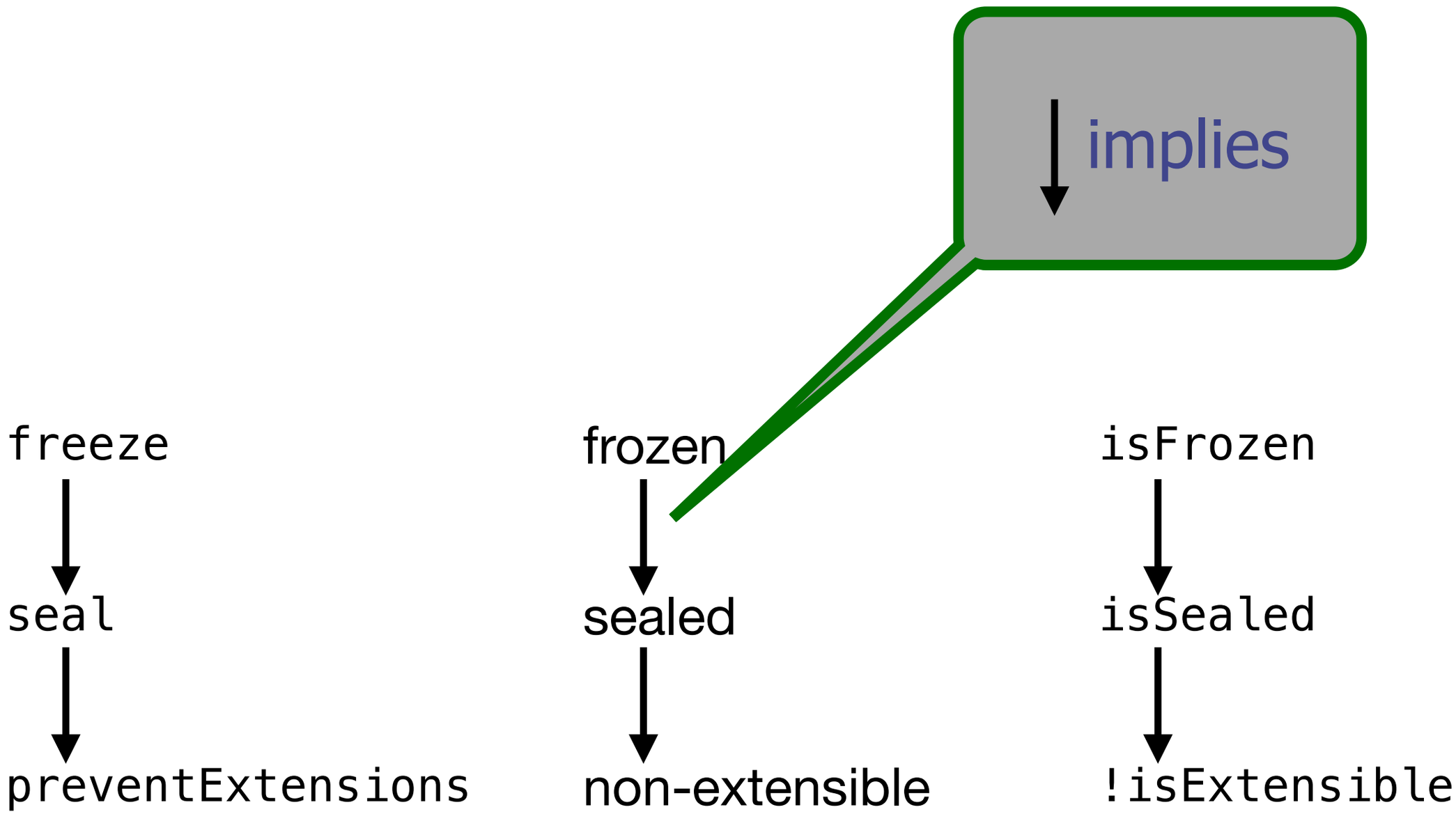
December 2024

TC 39

Recap: Current Integrity Traits

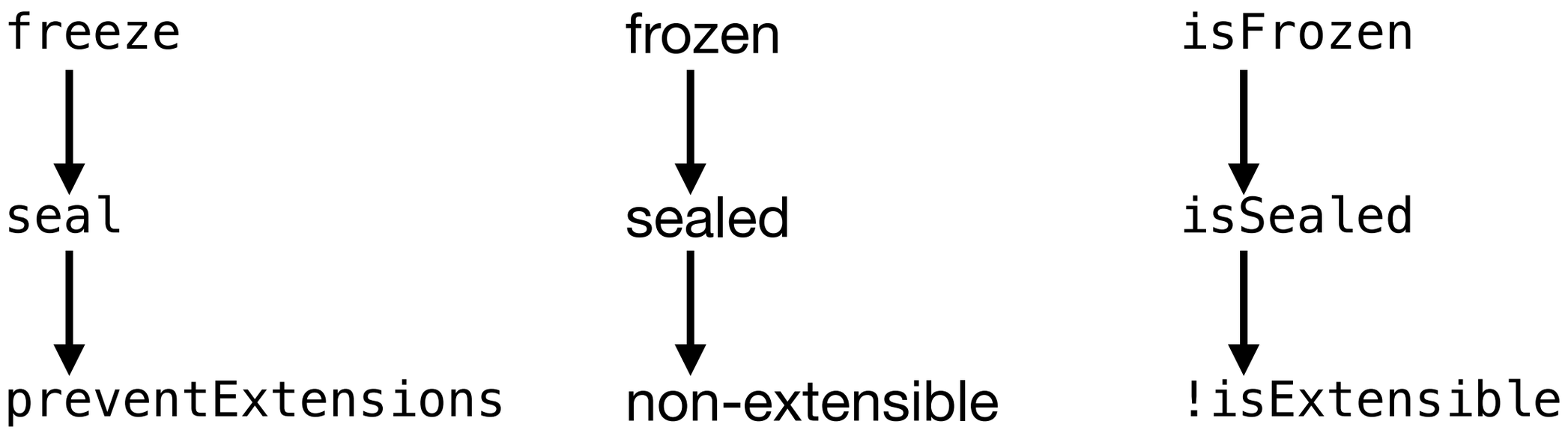


Recap: Current Integrity Traits



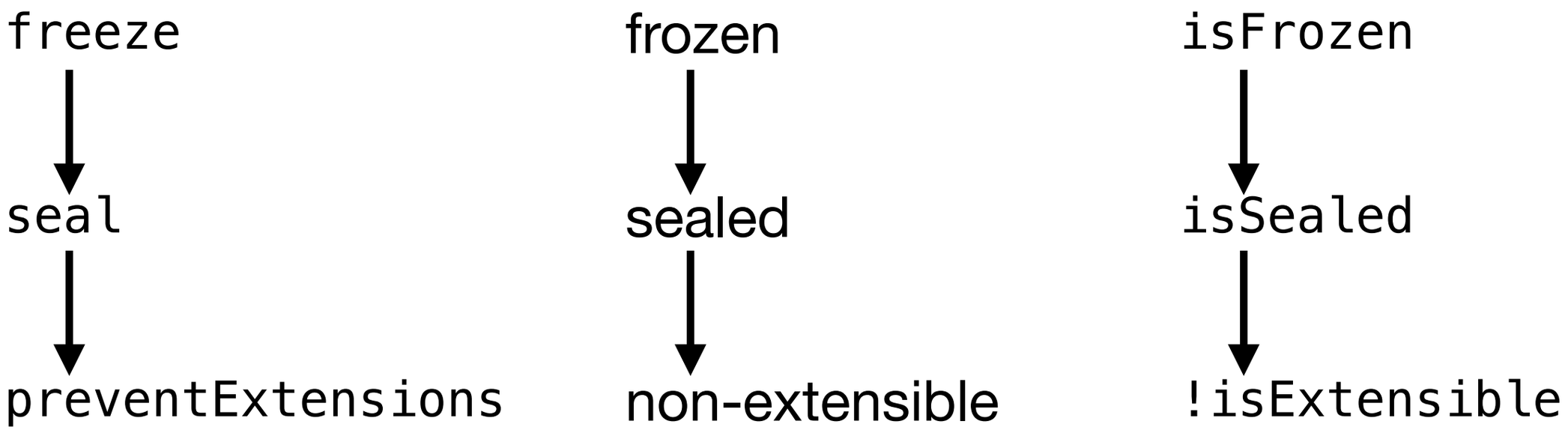
Recap: Current Integrity Traits

- Monotonic one-way switch



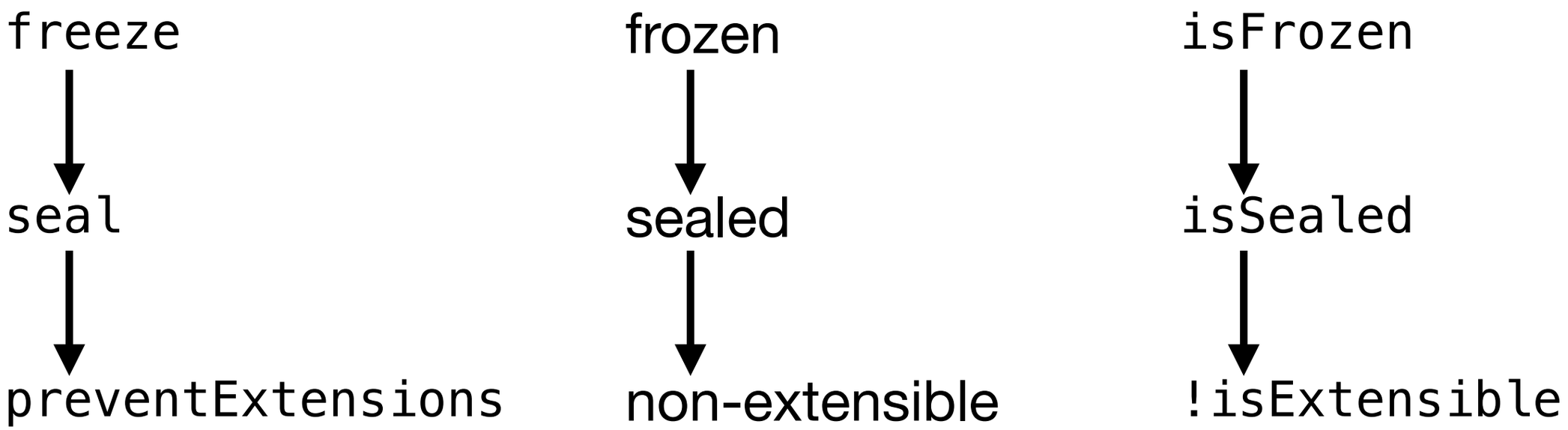
Recap: Current Integrity Traits

- Monotonic one-way switch
- Stronger object invariants

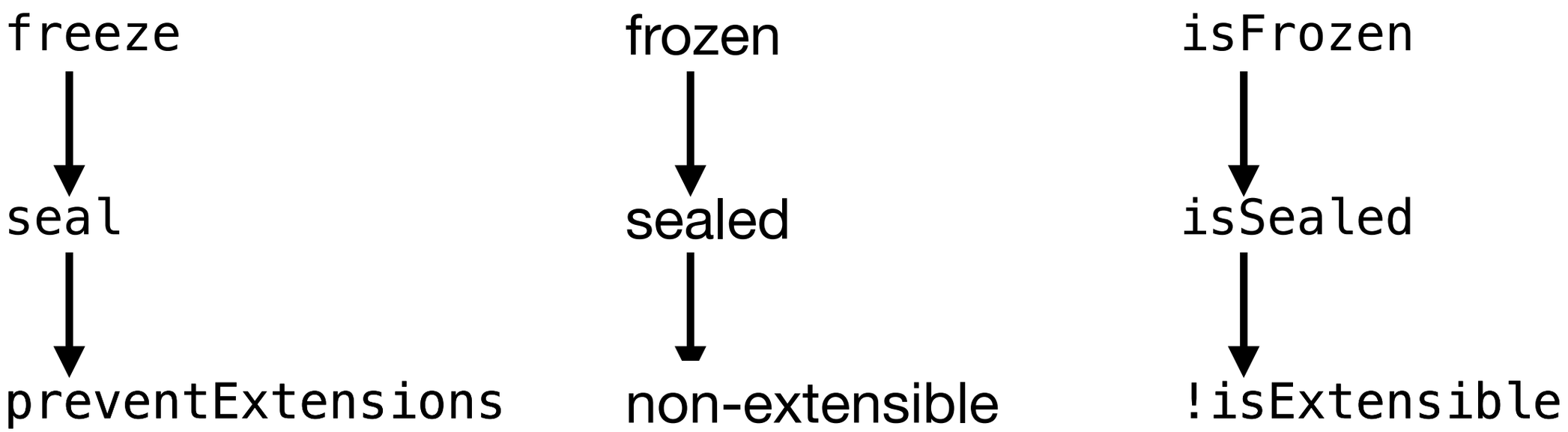


Recap: Current Integrity Traits

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

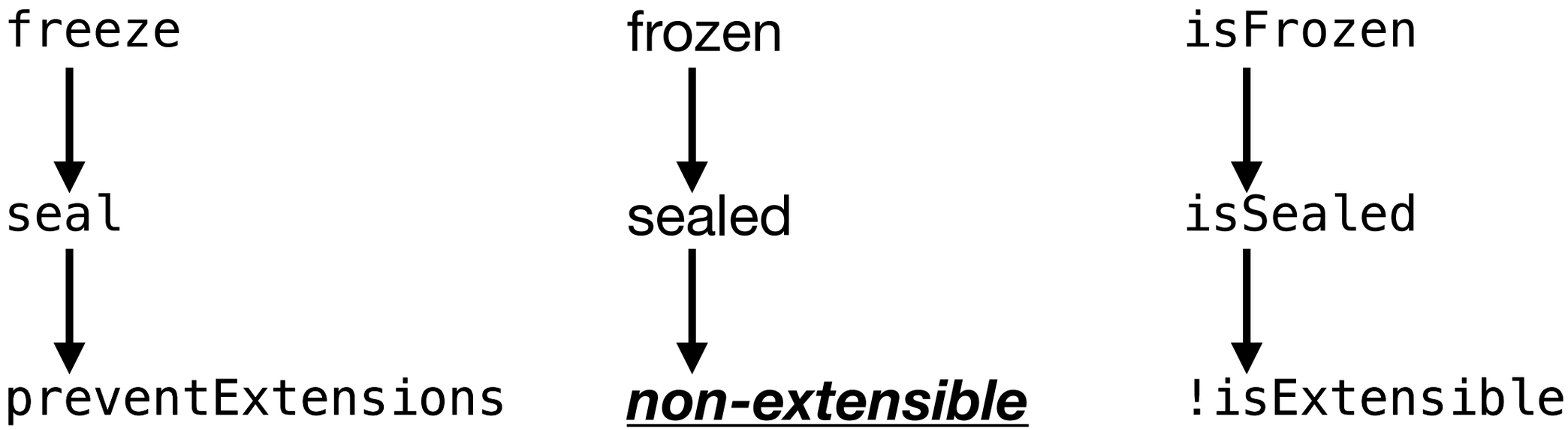


Recap: Current Integrity Traits



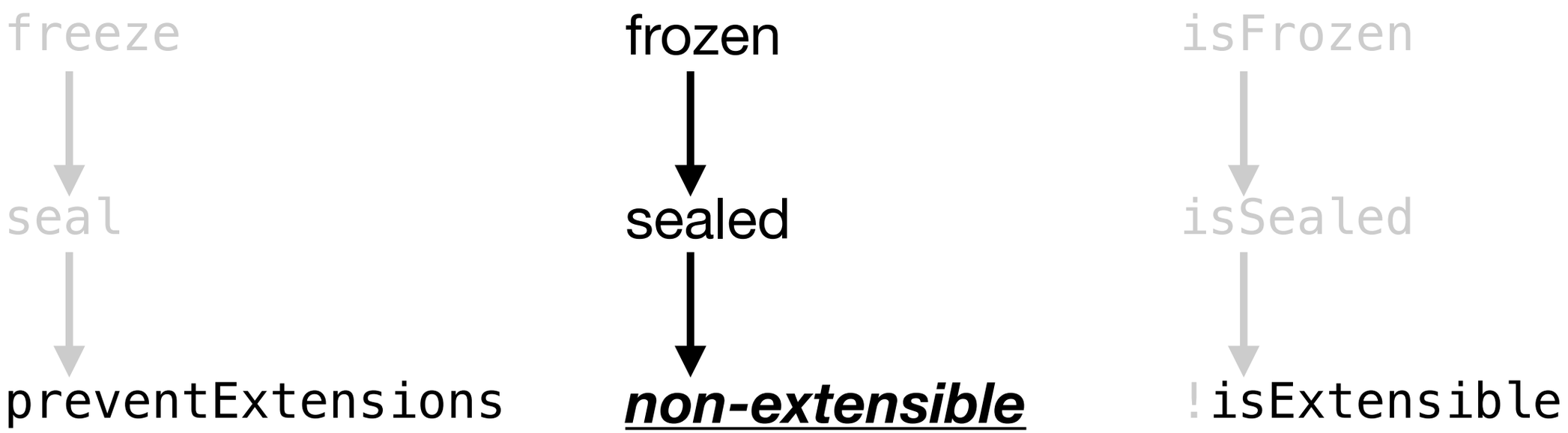
Recap: Current Integrity Traits

- ***Explicit*** vs Emergent



Recap: Current Integrity Traits

- ***Explicit*** vs Emergent
- 2 Proxy traps per ***explicit*** integrity trait



frozen

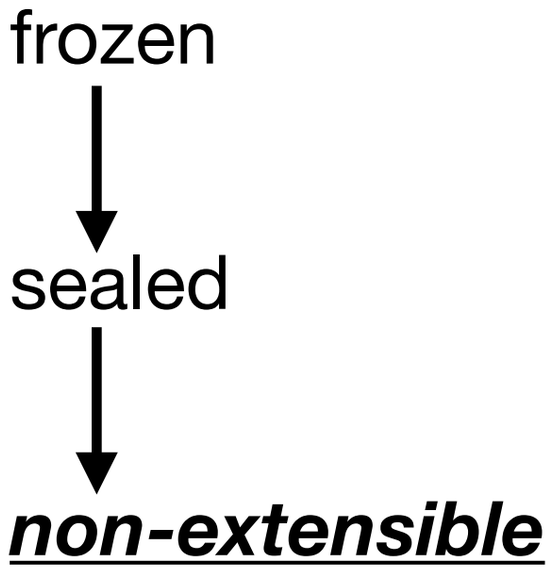
↓

sealed

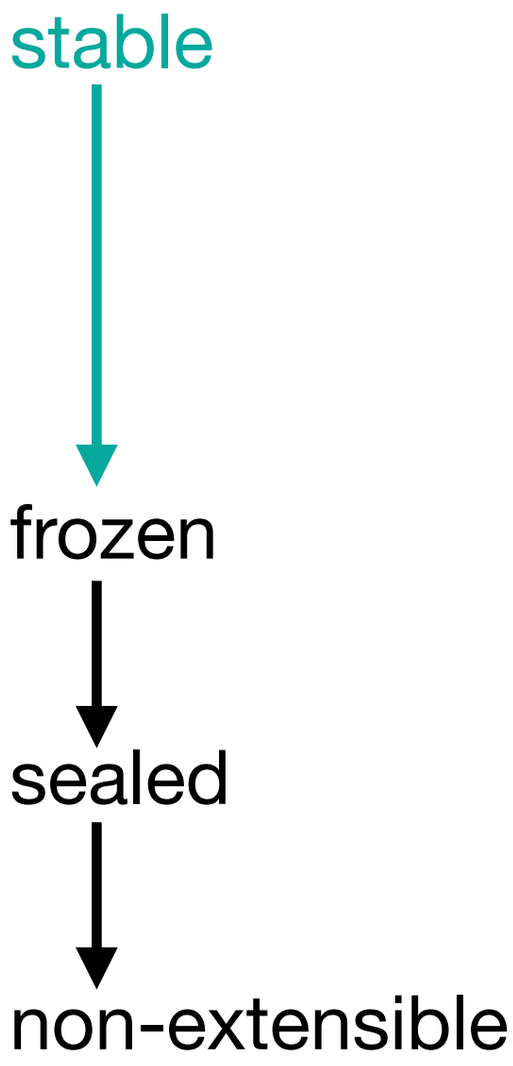
↓

non-extensible

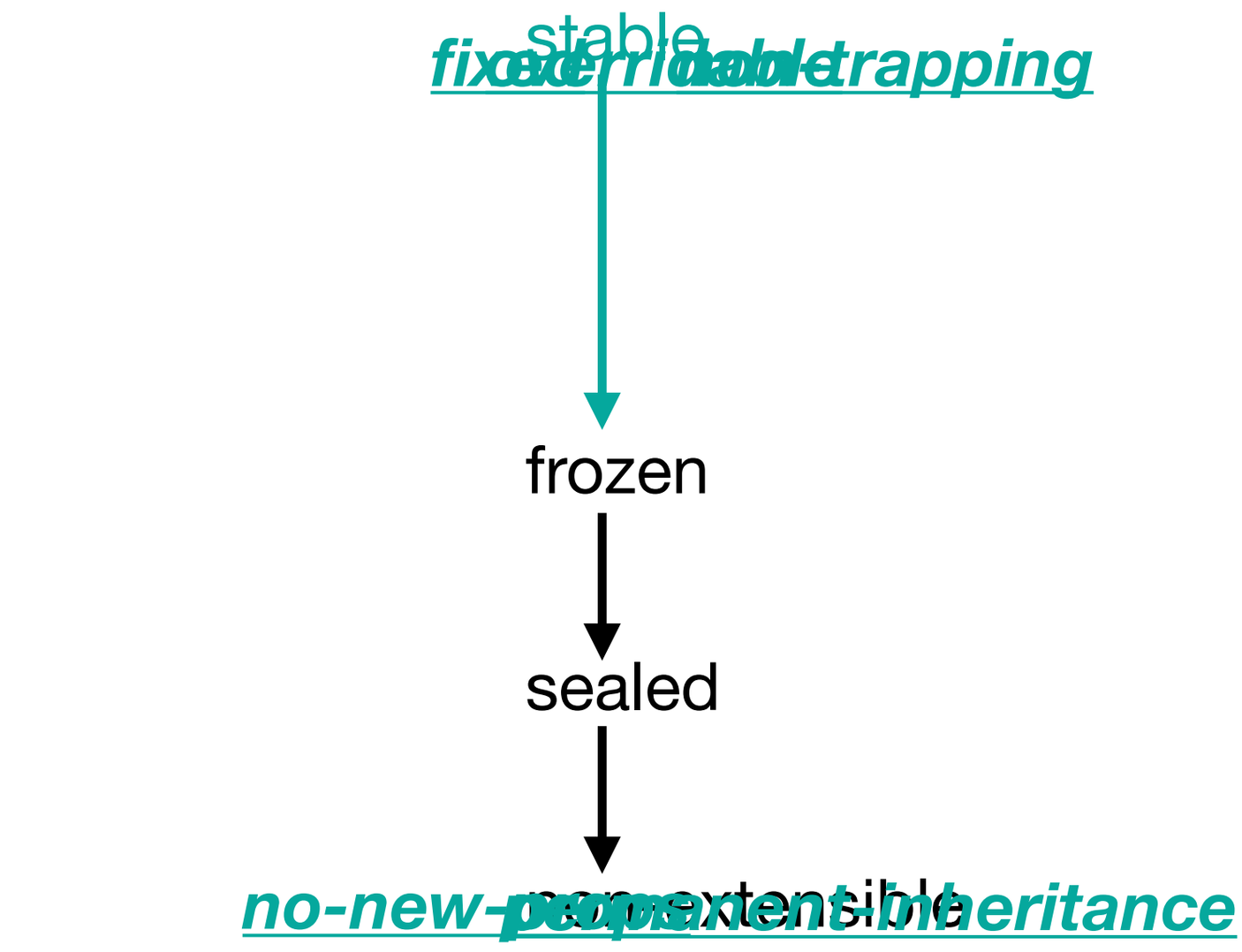
Recap: Unbundled Integrity Traits



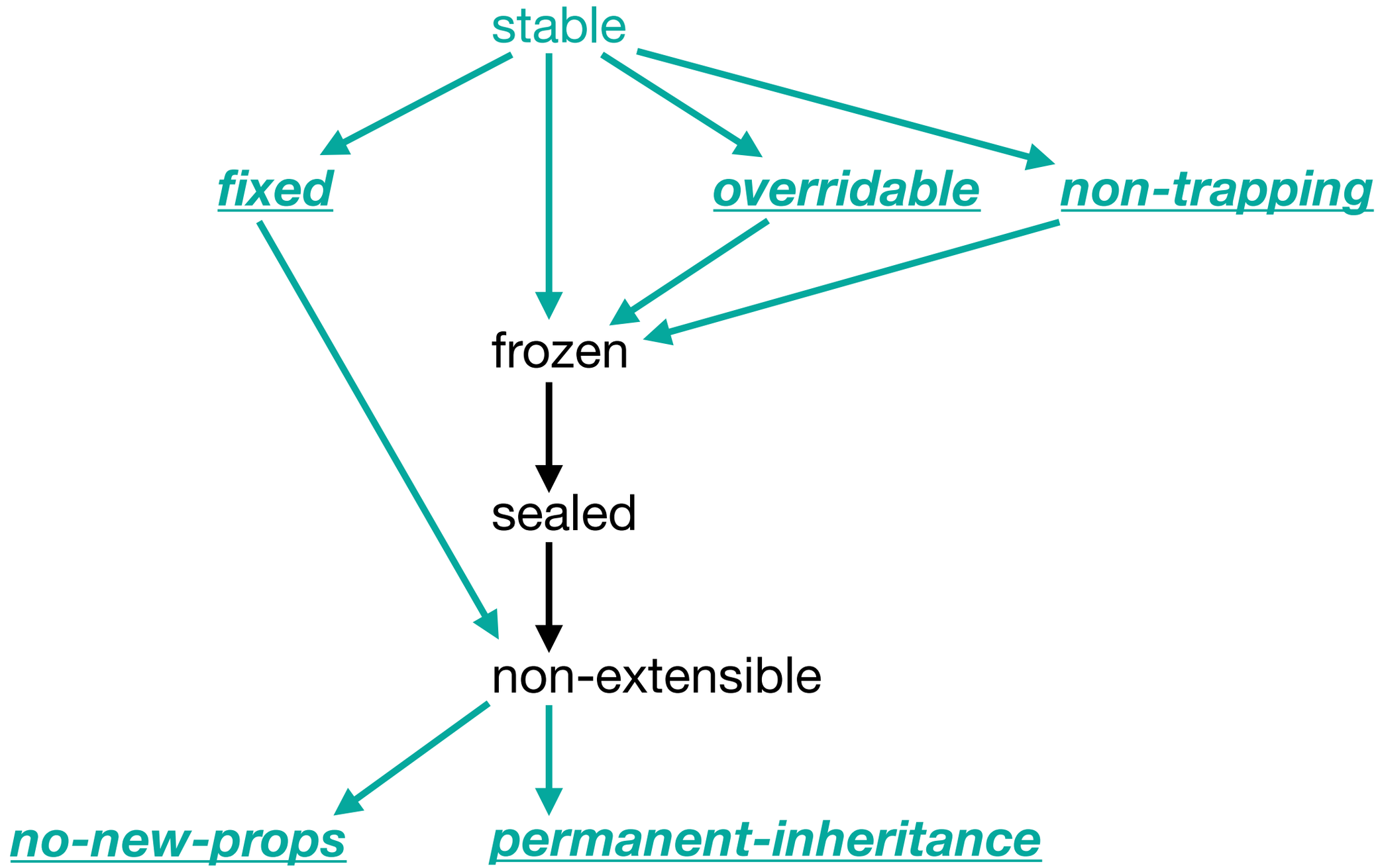
Recap: Unbundled Integrity Traits



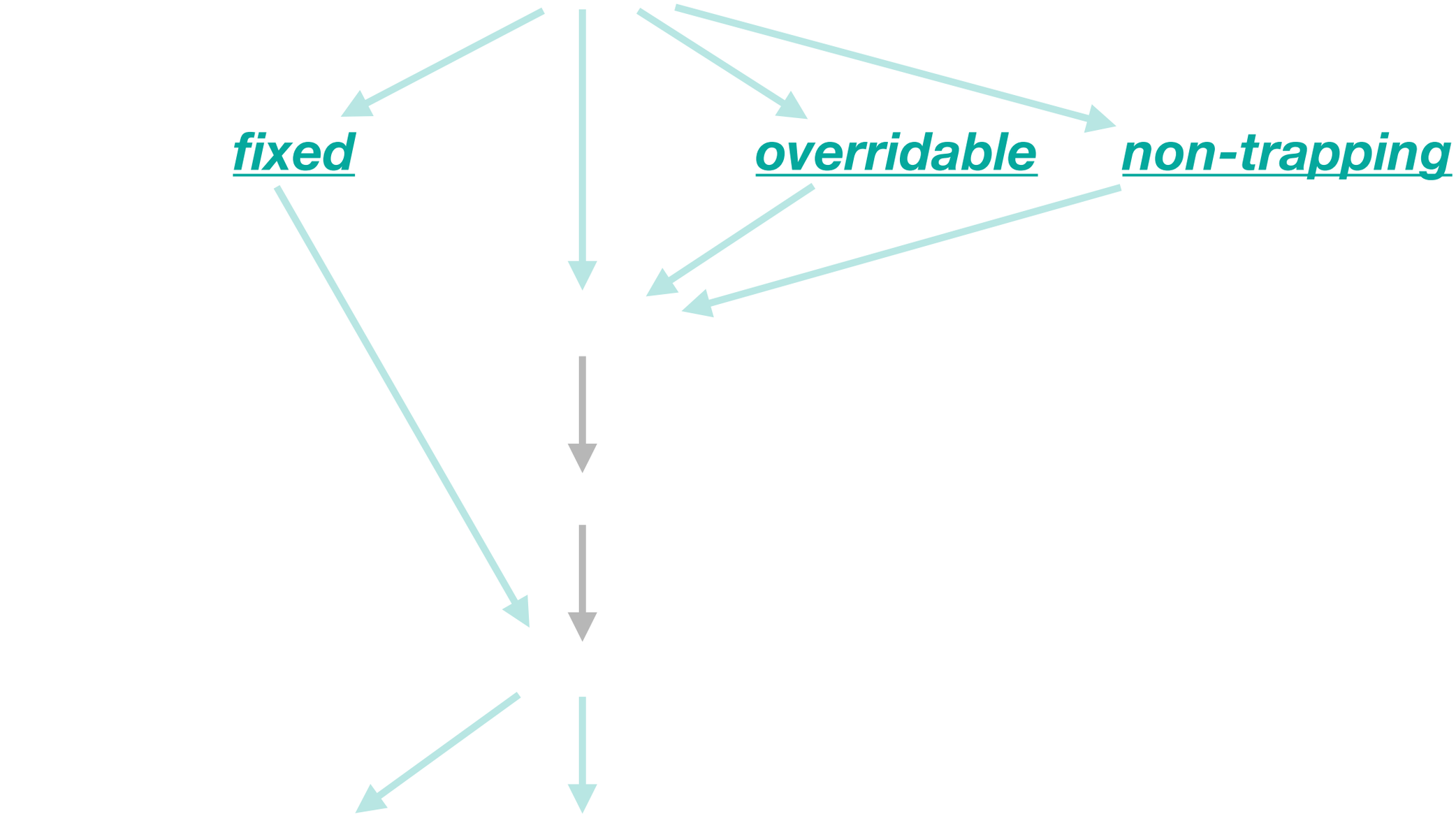
Recap: Unbundled Integrity Traits



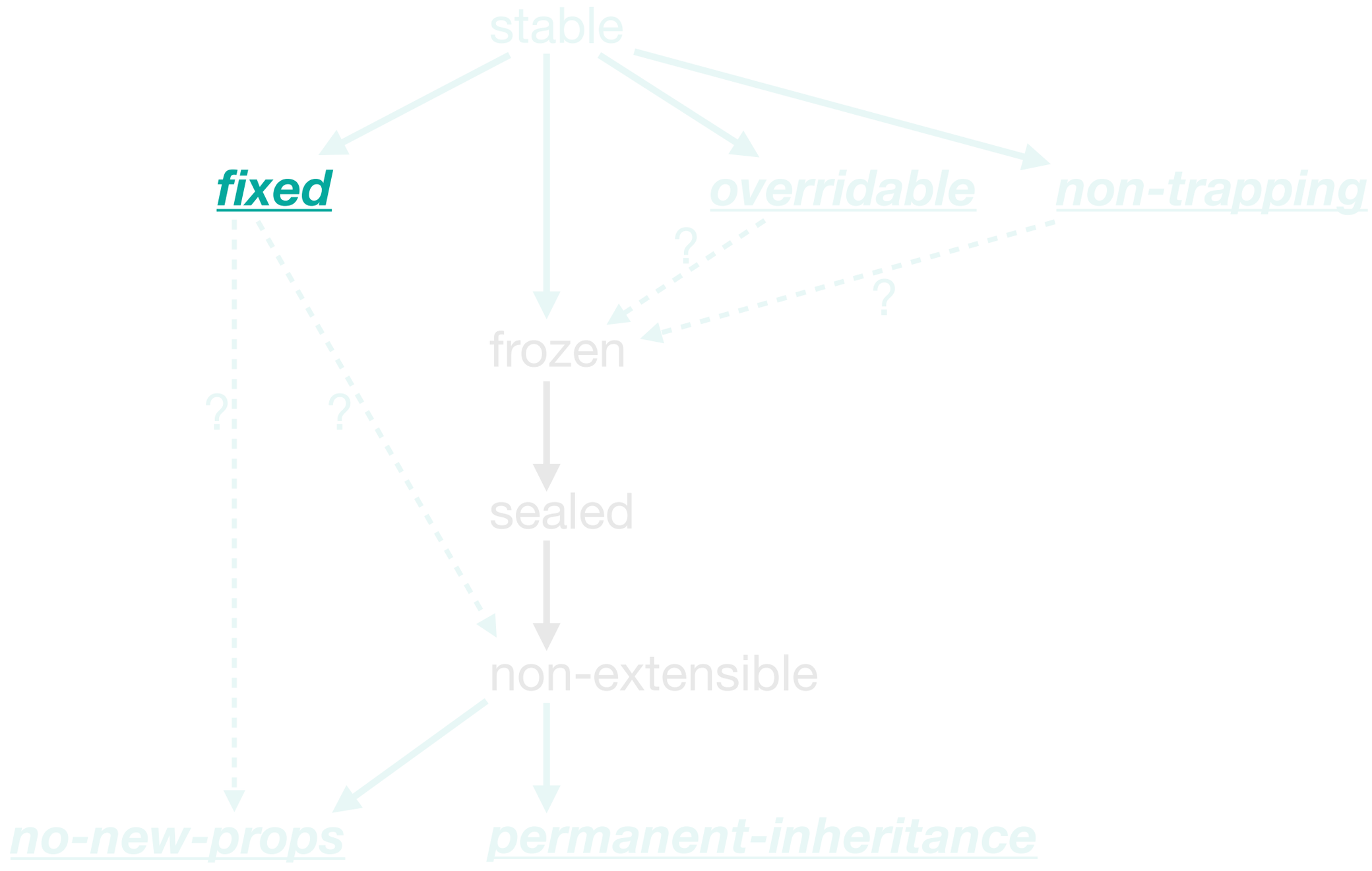
Recap: Unbundled Integrity Traits



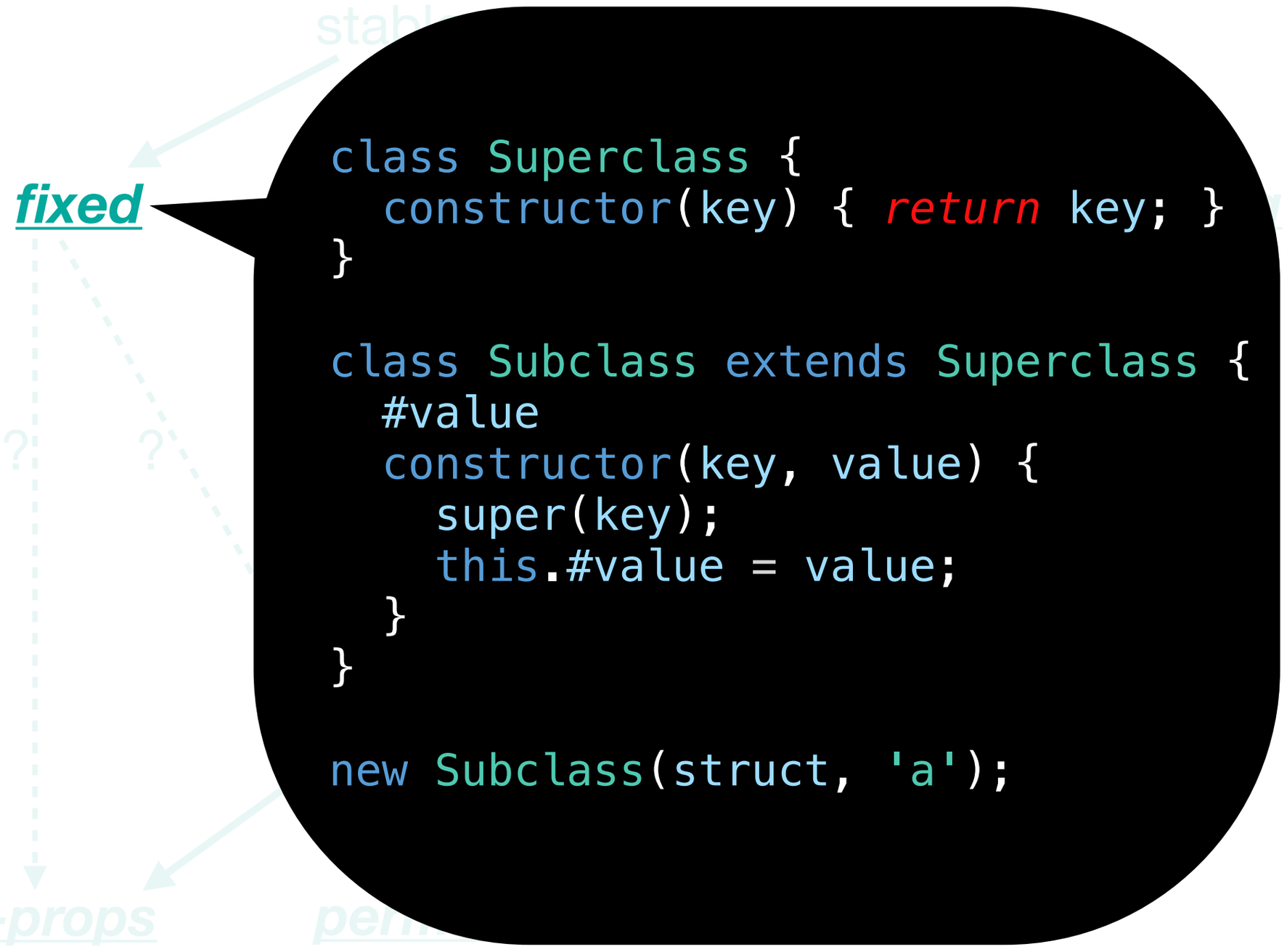
Recap: Unbundled Integrity Traits



Recap: ***fixed***
Mitigate return-override mistake



Recap: ***fixed***
Mitigate return-override mistake



Recap: ***fixed***
Mitigate return-override mistake

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 *stamping*

Fixed shape structs

Retcon `windowProxy`

Virtualizable weakness

V8 prefers to normatively change non-extensible to imply fixed (no private stamping) #5

Open

syg opened on Dec 9, 2024 · edited by syg

Edits

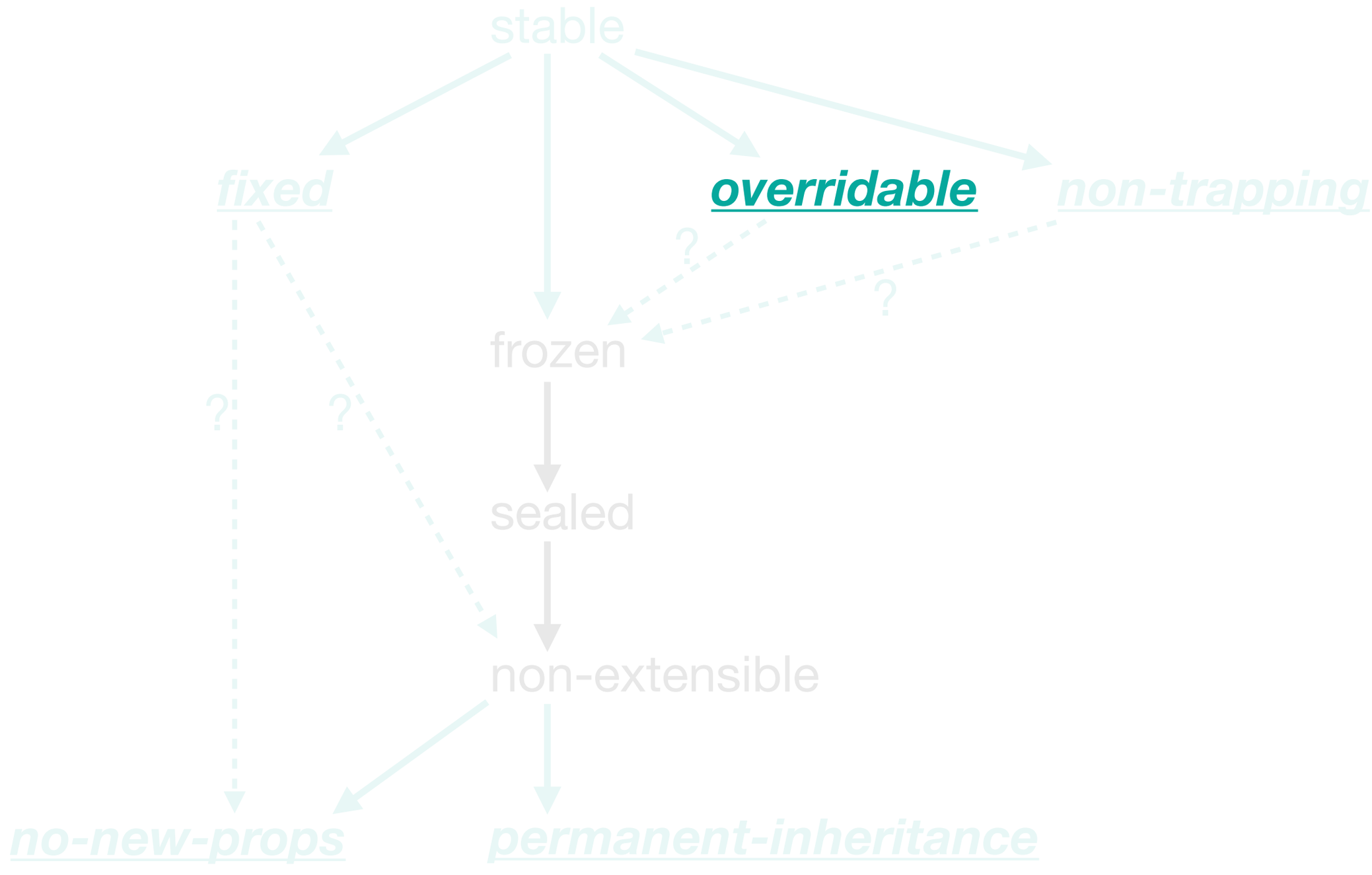
V8 team's preference is to remove the need for an additional bit for the "no private stamping" behavior, and instead to change non-extensible to imply it. We think this is most natural and aligns best with programmer intuition.

The big question is whether this change is backwards compatible. To that end we've added the use counter `V8ExtendingNonExtensibleWithPrivate` to see if anyone is actually adding private fields to non-extensible objects today. This is scheduled to hit Chrome stable builds by the end of January, so we won't see truly representative numbers until next year.

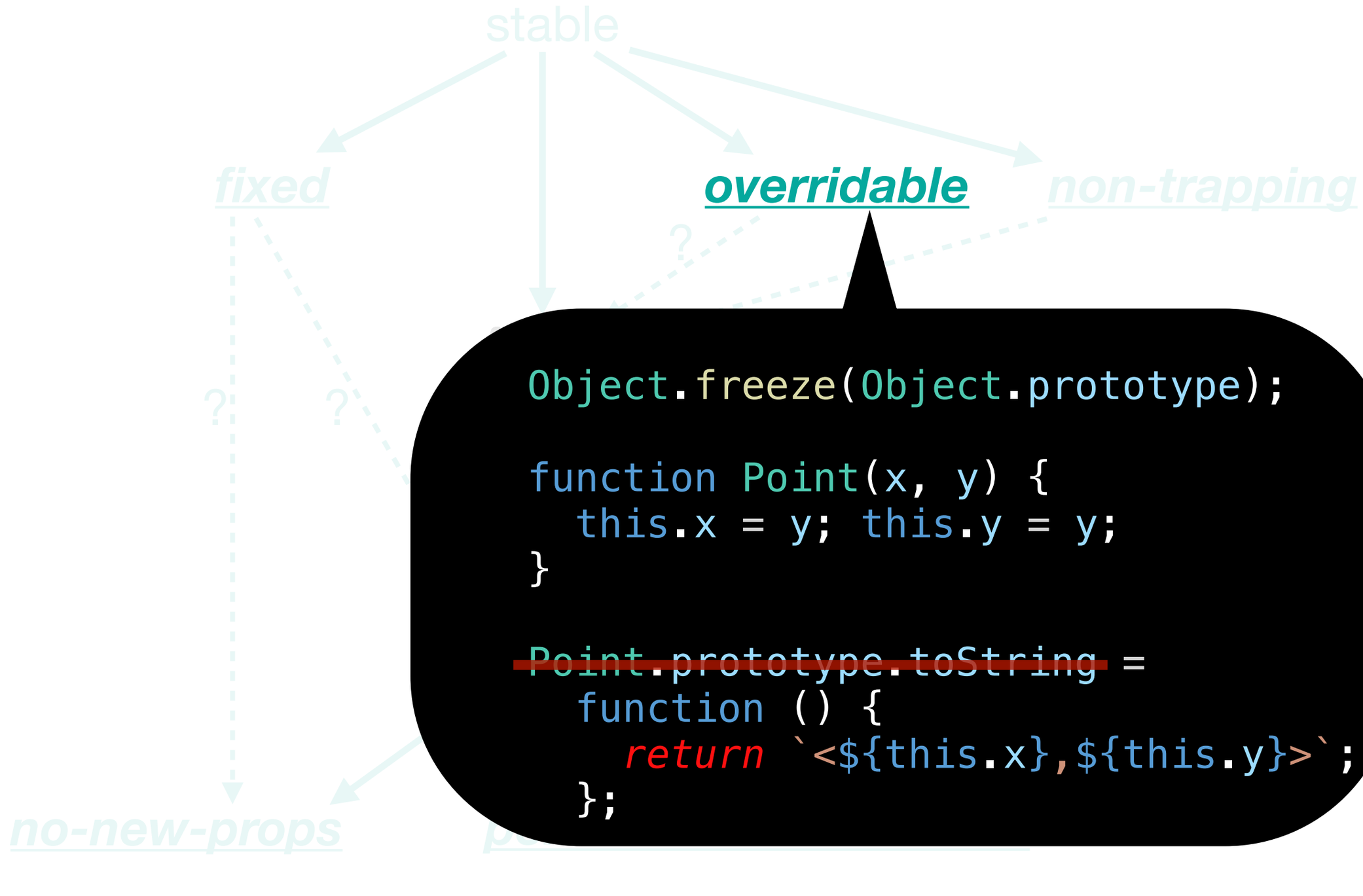
In the December 2024 TC39, other delegates and champions were positive on the idea of this change if it's compatible to make.

Create sub-issue

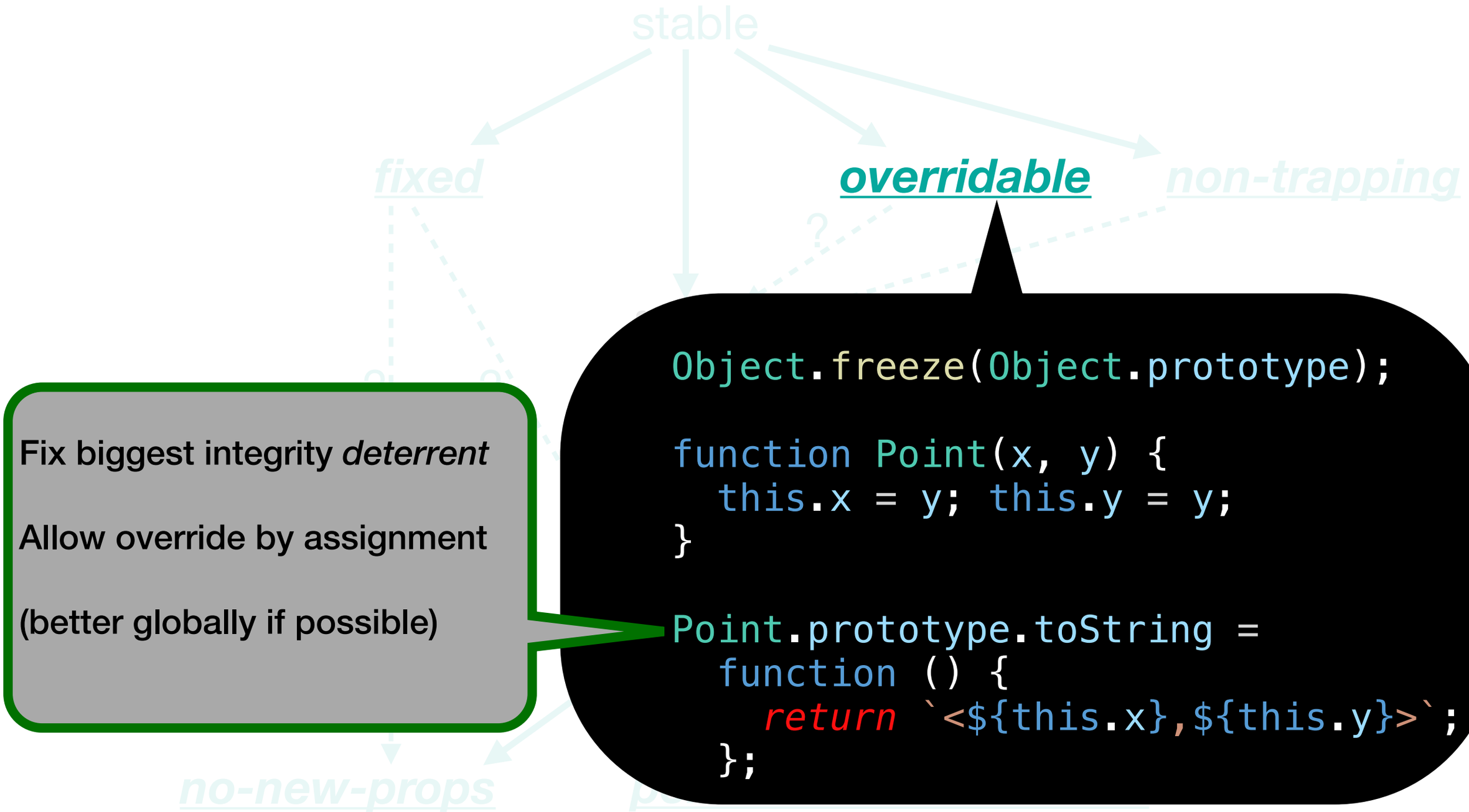
Recap: ***overridable***
Mitigate assignment-override mistake



Recap: ***overridable***
Mitigate assignment-override mistake



Recap: ***overridable***
Mitigate assignment-override mistake



Possible solution to override mistake without a new integrity trait #4

Open

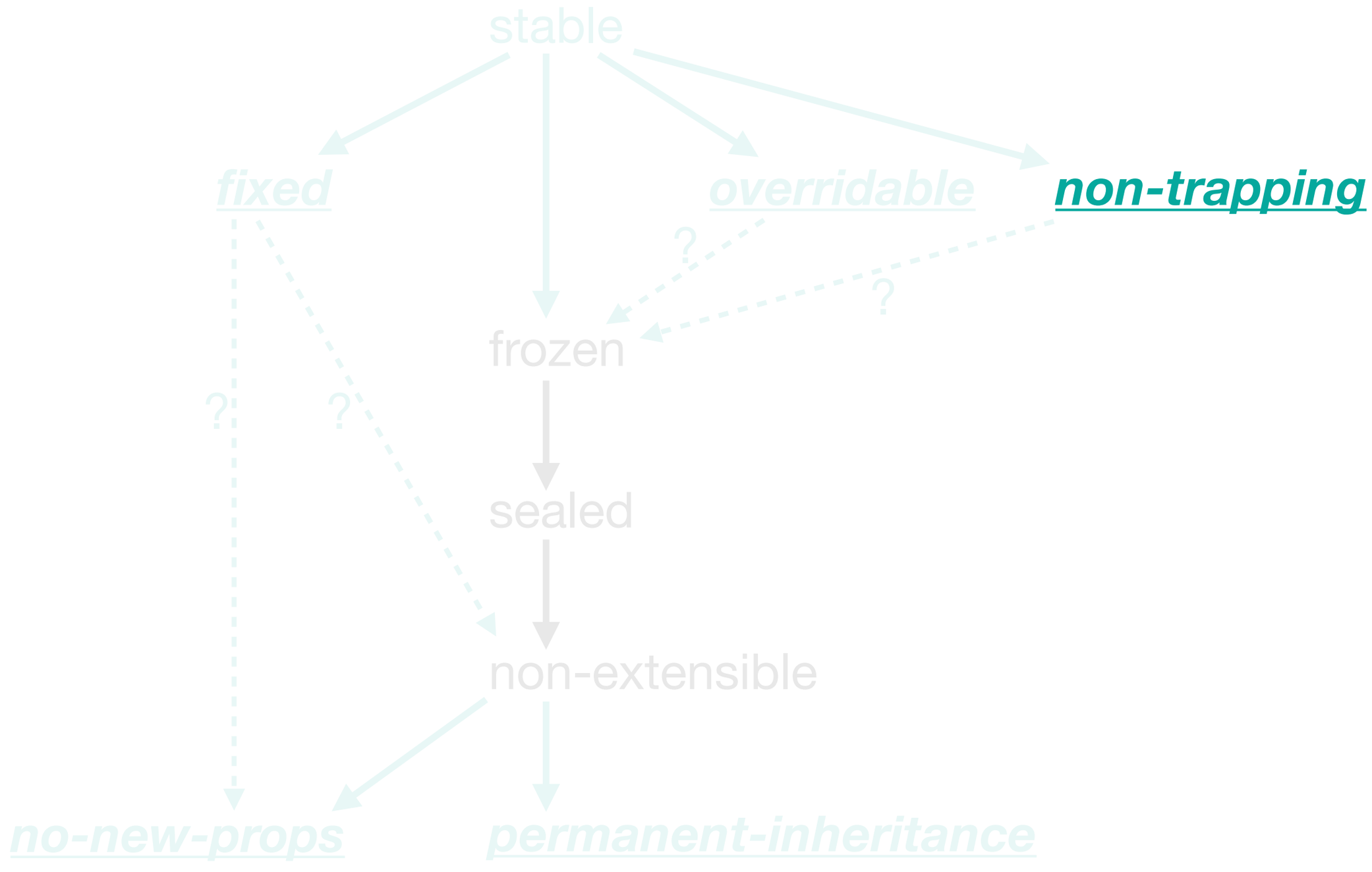
jridgewell opened on Dec 3, 2024 · edited by jridgewell Edits •••

Based on the bug found in tc39/ecma262#1320 (comment):

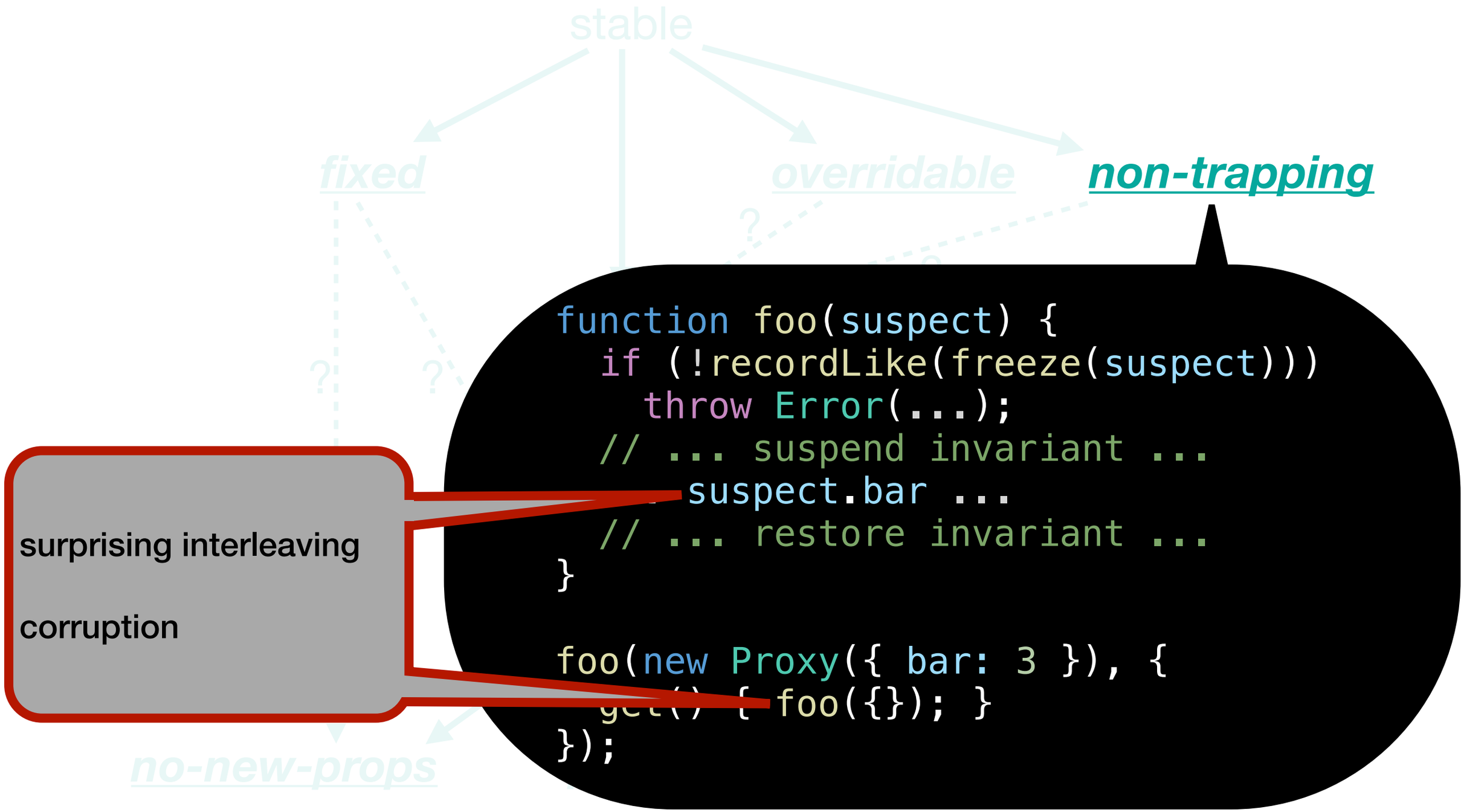
1. If a non-writable prototype property exists during `o.foo = v`
    i. Define a new property on `o` with `{ value: v, enumerable: true, writable: true, configurable: true }` (like it's defining a new property without extending the parent)
    ii. Note: It must be configurable to get past the `delete` statement when running `getRawTag(new Uint8Array(0))`.
2. Extend `Object.prototype.toString()` special cases with typed arrays, `DataView`, `ArrayBuffer`, `Map`, `WeakMap`, `Set`, `WeakSet`, `Promise`, etc.
    i. Note: With the ability to write `value[Symbol.toStringTag] = undefined`, we must fix various builtins to still return the expected `[object Foo]` values.
    ii. Note: This fixes lodash's `isTypedArray`, `isArrayBuffer`, etc, functions directly, and the `getTag` function indirectly by ensuring a few feature checks don't fail and cause a incomplete impl of `getTag` to be used.

Create sub-issue ▾ ☺

Recap: ***non-trapping***
Mitigate proxy reentrancy hazards



Recap: ***non-trapping***
Mitigate proxy reentrancy hazards



Recap: ***non-trapping***
Mitigate proxy reentrancy hazards

records were just pojos

No handler traps
(or equivalent for exotics)

pojos help defensiveness

stable

overridable

non-trapping

no-new-props

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

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

feat(non-trapping-shim): opt-in shim of the non-trapping integrity trait #2673

Open

Conversation 39 | Commits 8 | Checks 14 | Files changed 19

erights commented on Dec 29, 2024 • edited ▾ Member ⋯

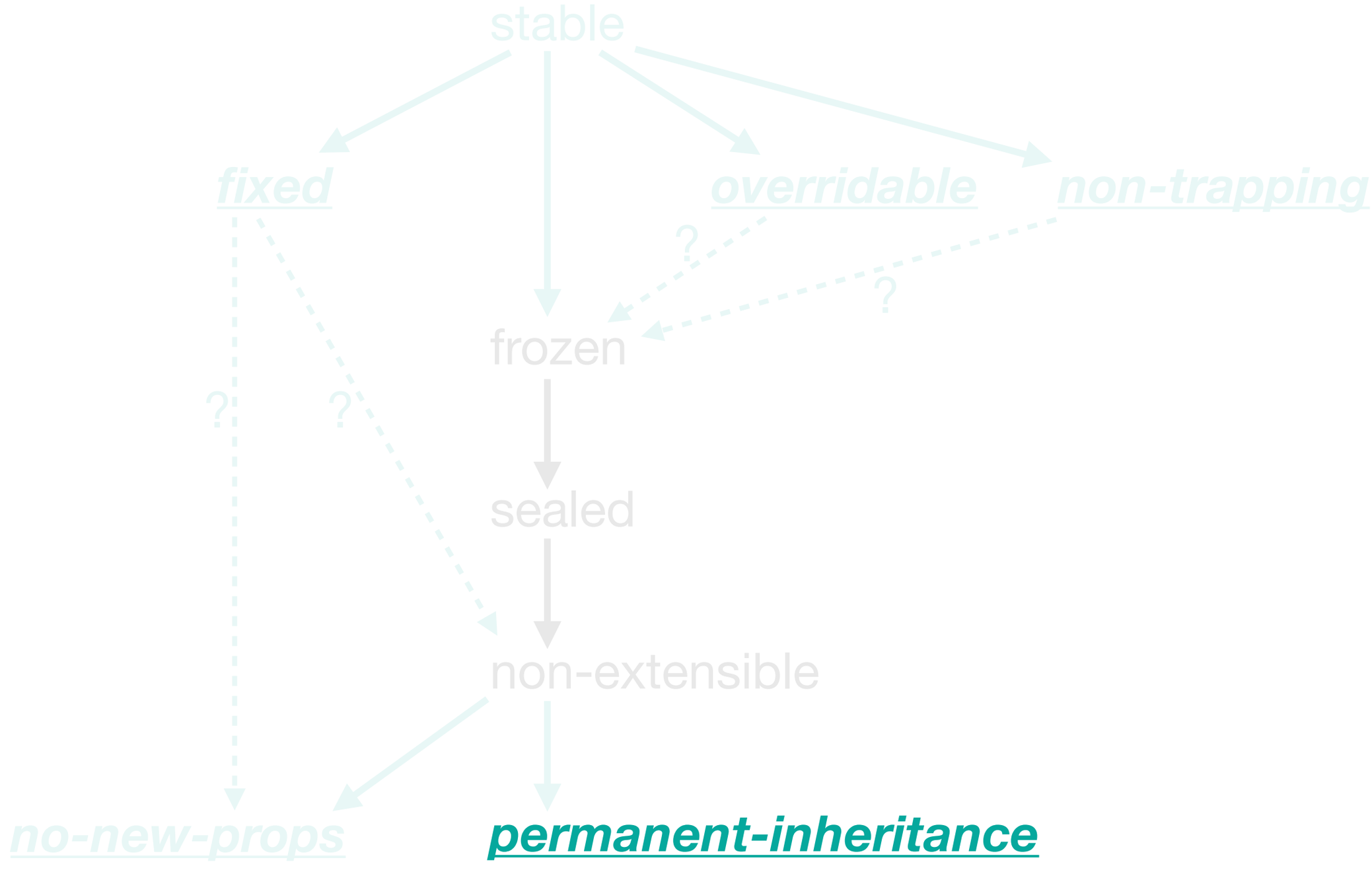
Description

An opt-in shim for the non-trapping integrity level of https://github.com/tc39/proposal-stabilize

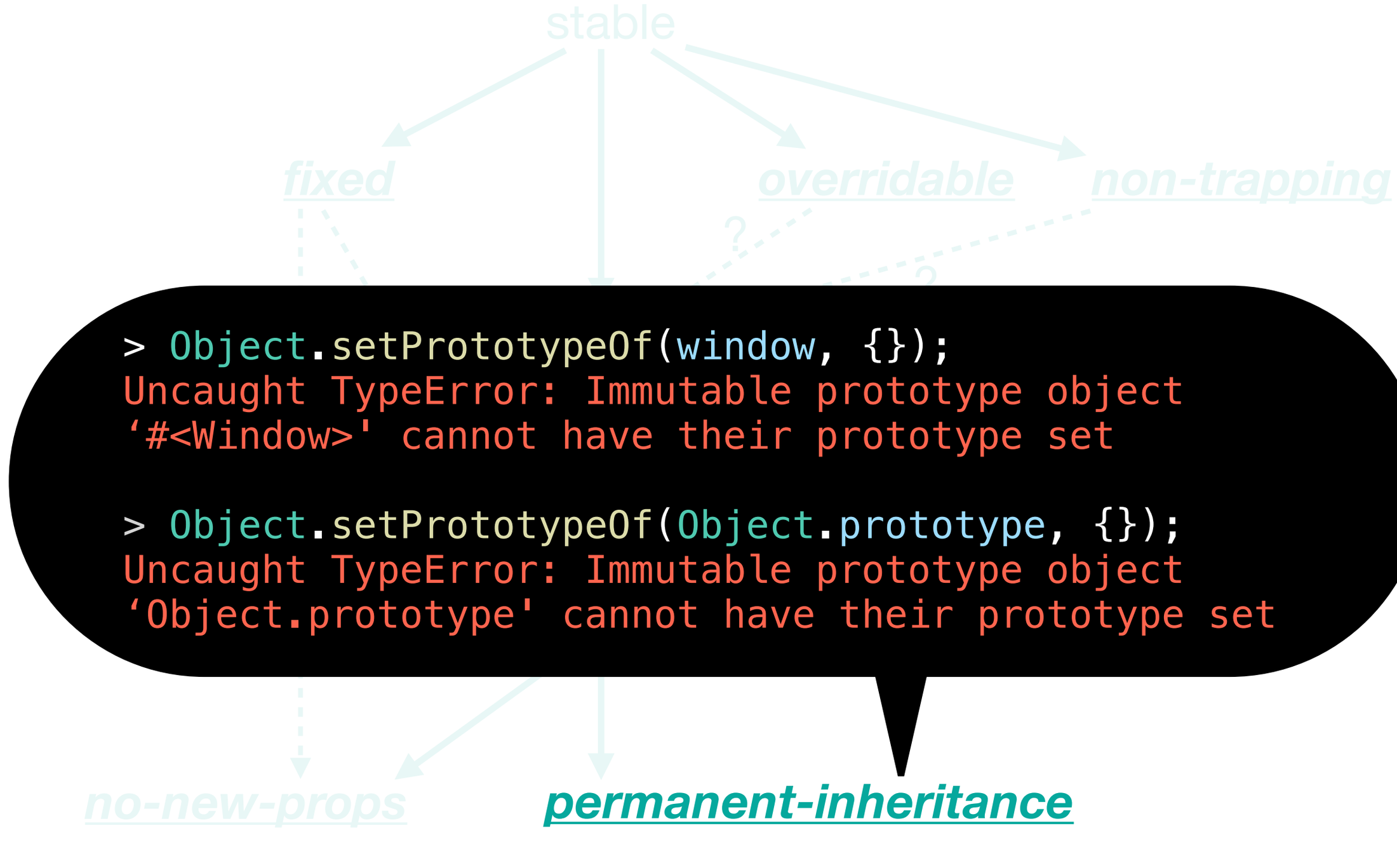
This does the non-trapping trait by itself only implying frozen, since we cannot yet ponyfill or shim "fix" or "permit-overrides". However, non-trapping would actually need to imply all other integrity levels, since the methods for querying or enabling these integrity levels would no longer trap once an object is non-trapping.

Recap: ***permanent-inheritance***

Retcon `windowProxy`, `Object.prototype`



Recap: ***permanent-inheritance***
Retcon `windowProxy`, `Object.prototype`



Recap: ***permanent-inheritance***
Retcon `windowProxy`, `Object.prototype`

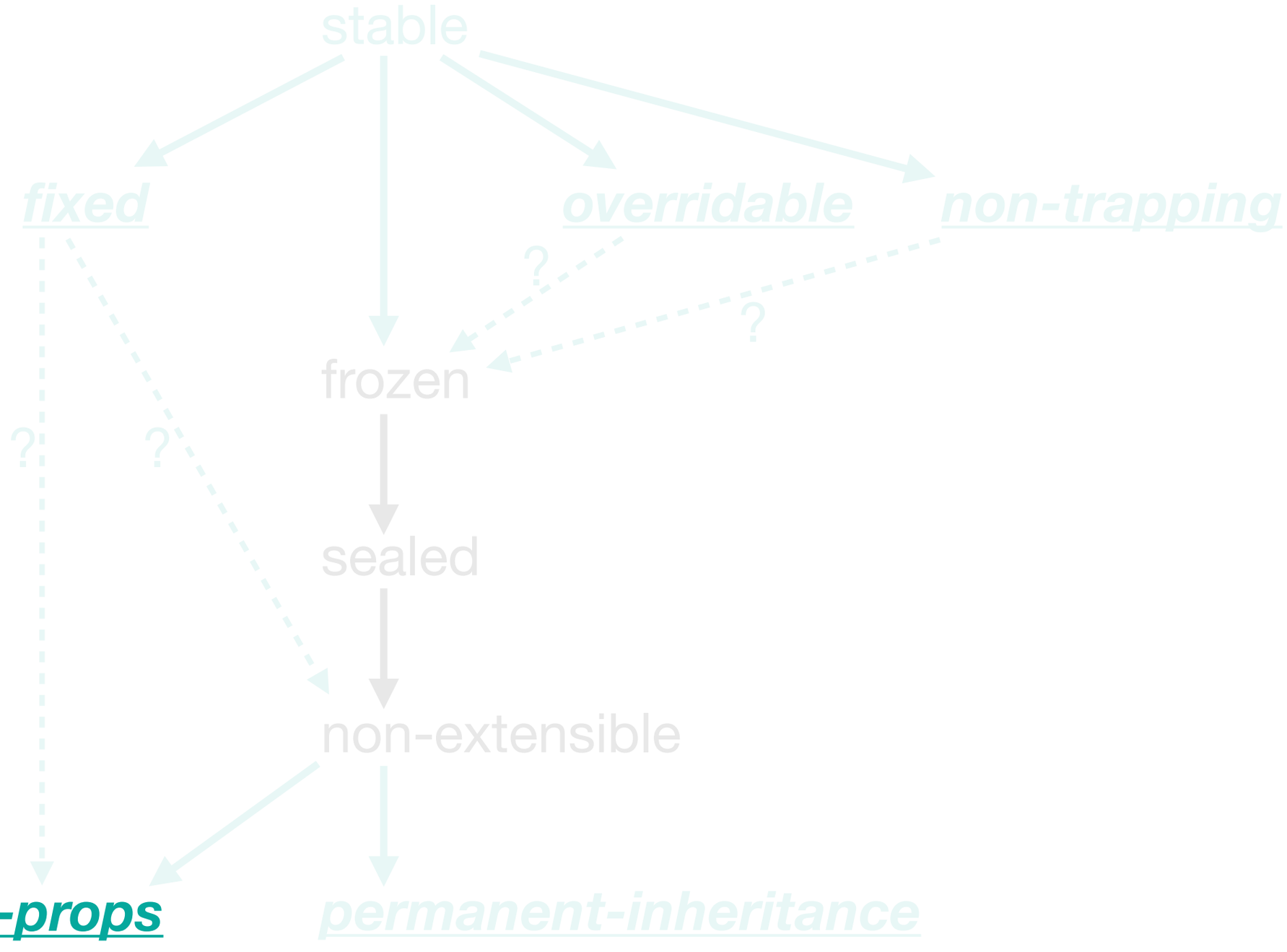
Too much magic

Make objects less exotic

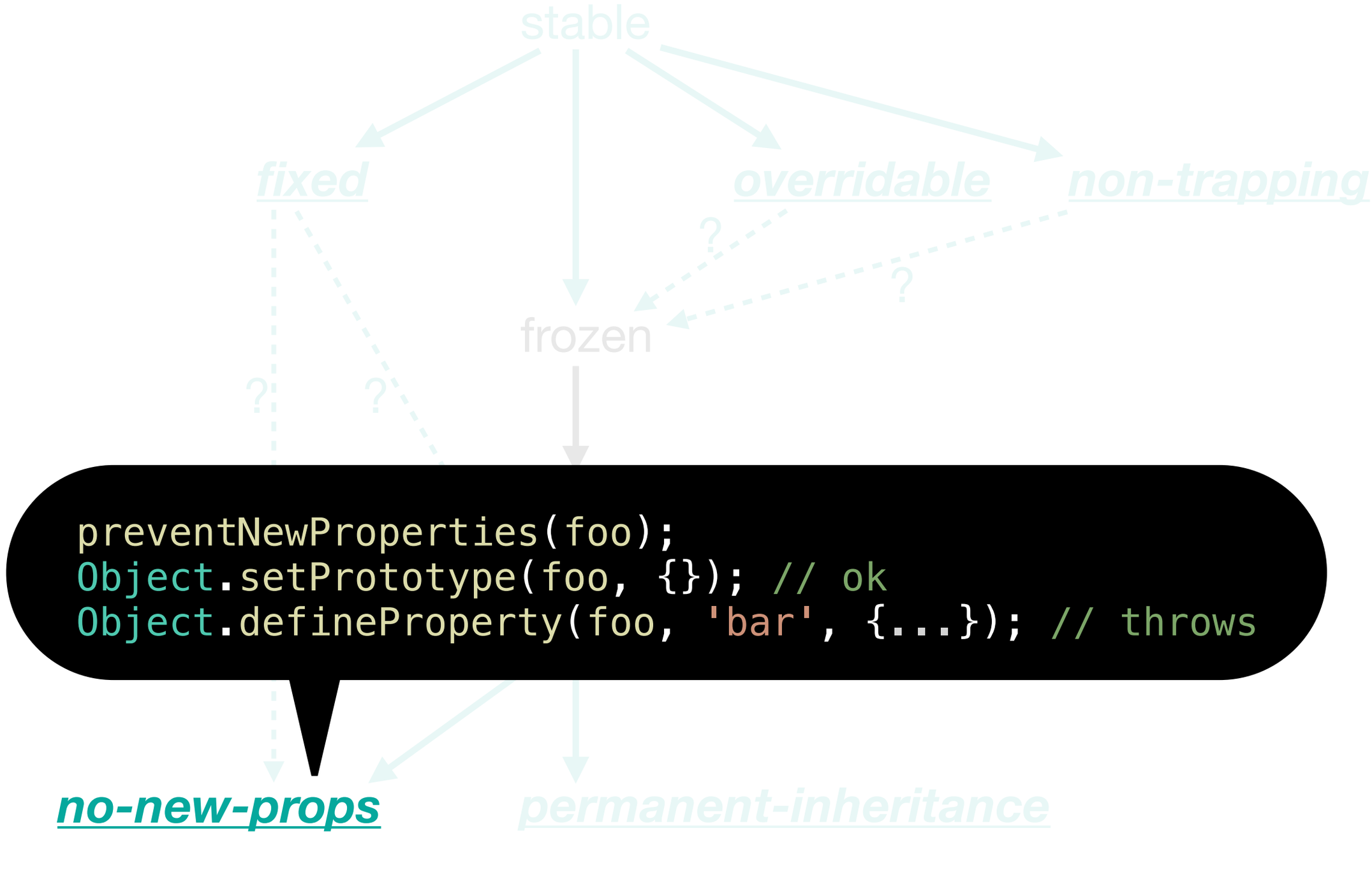
```
> Object.setPrototypeOf(window, {});
Uncaught TypeError: Immutable prototype object
‘#<Window>' cannot have their prototype set

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

fixed *non-trapping* *no-new-props* ***permanent-inheritance***

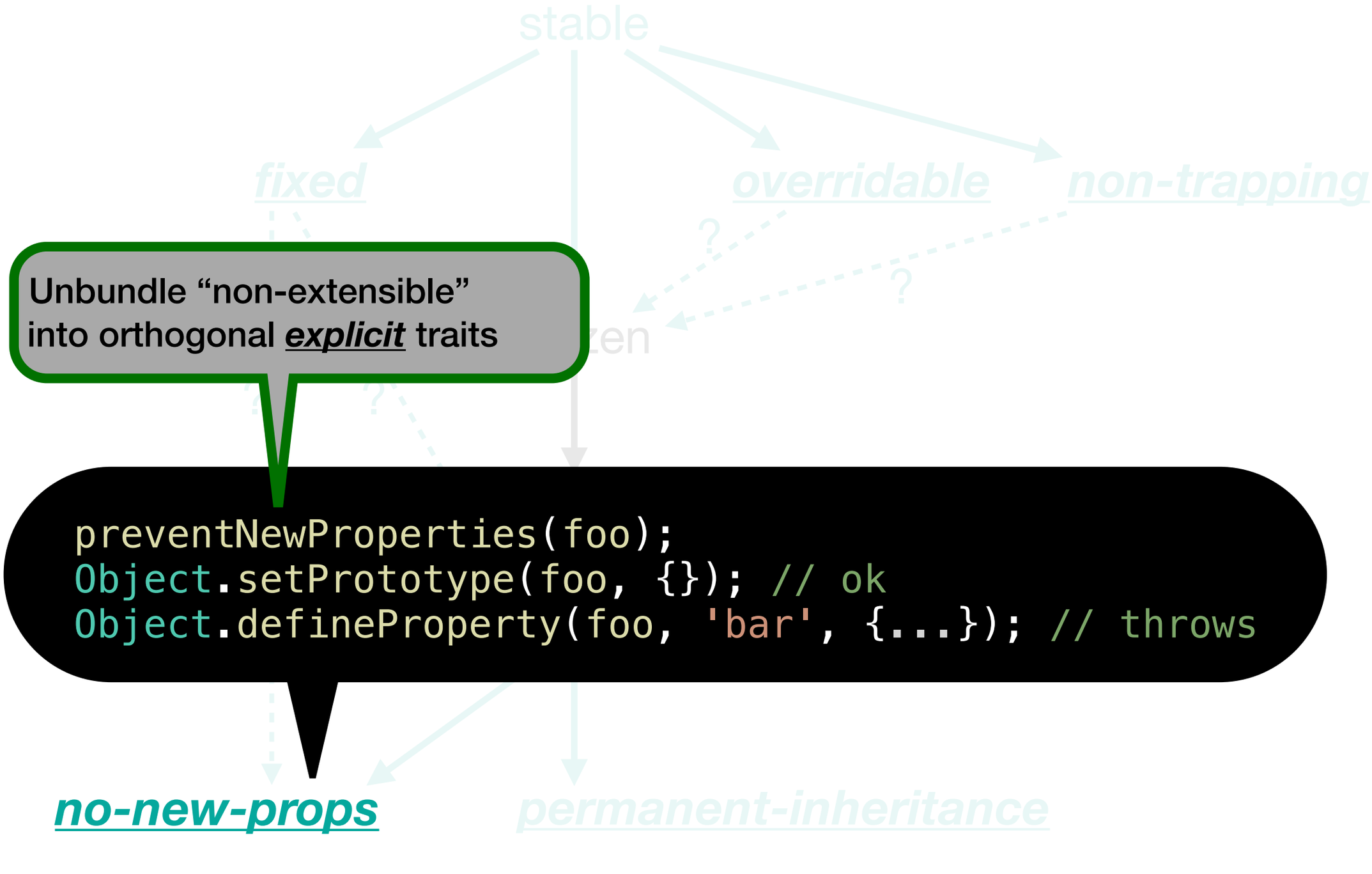
Recap: ***no-new-props***
Rest of non-extensible unbundling

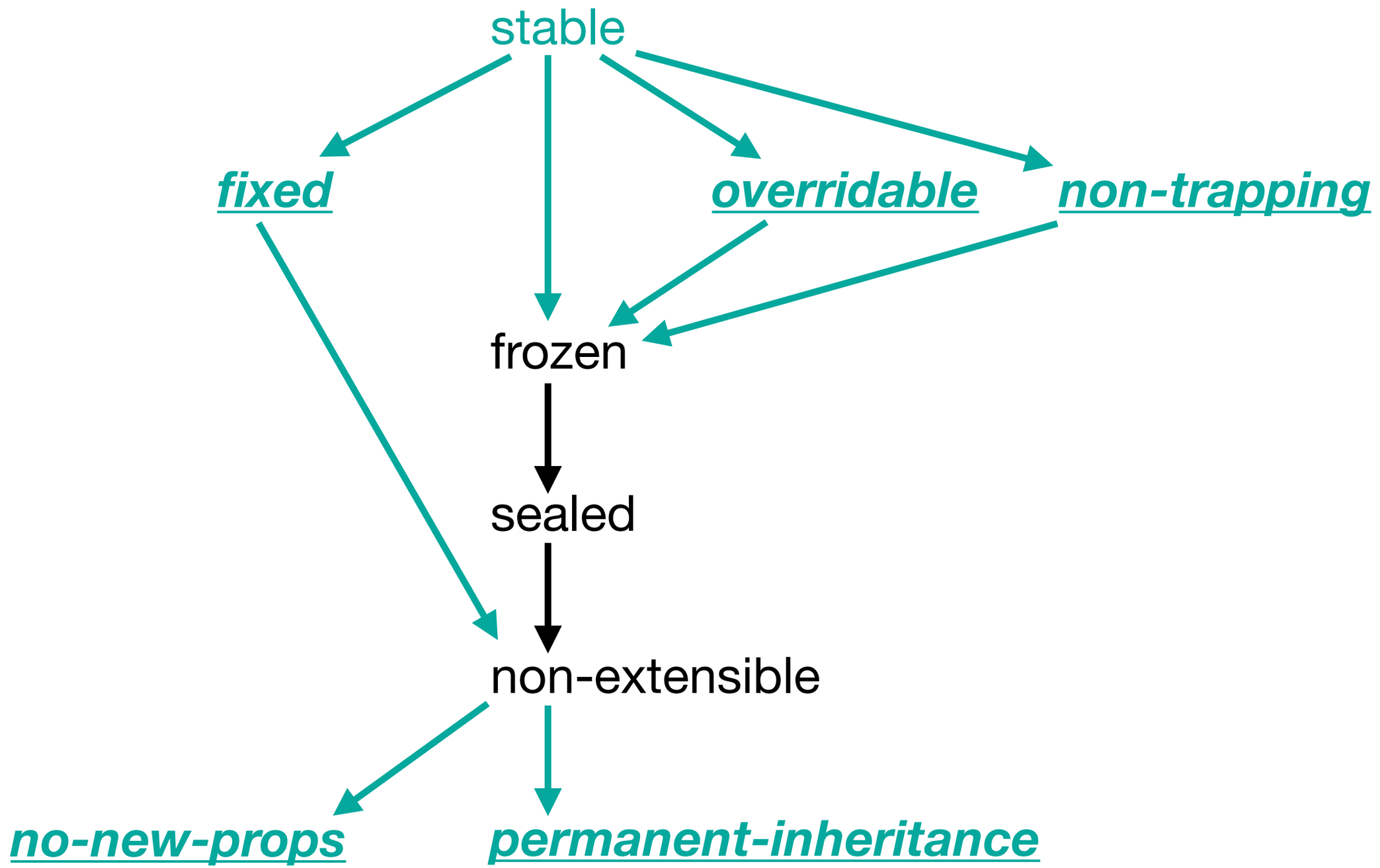


Recap: ***no-new-props***
Rest of non-extensible unbundling

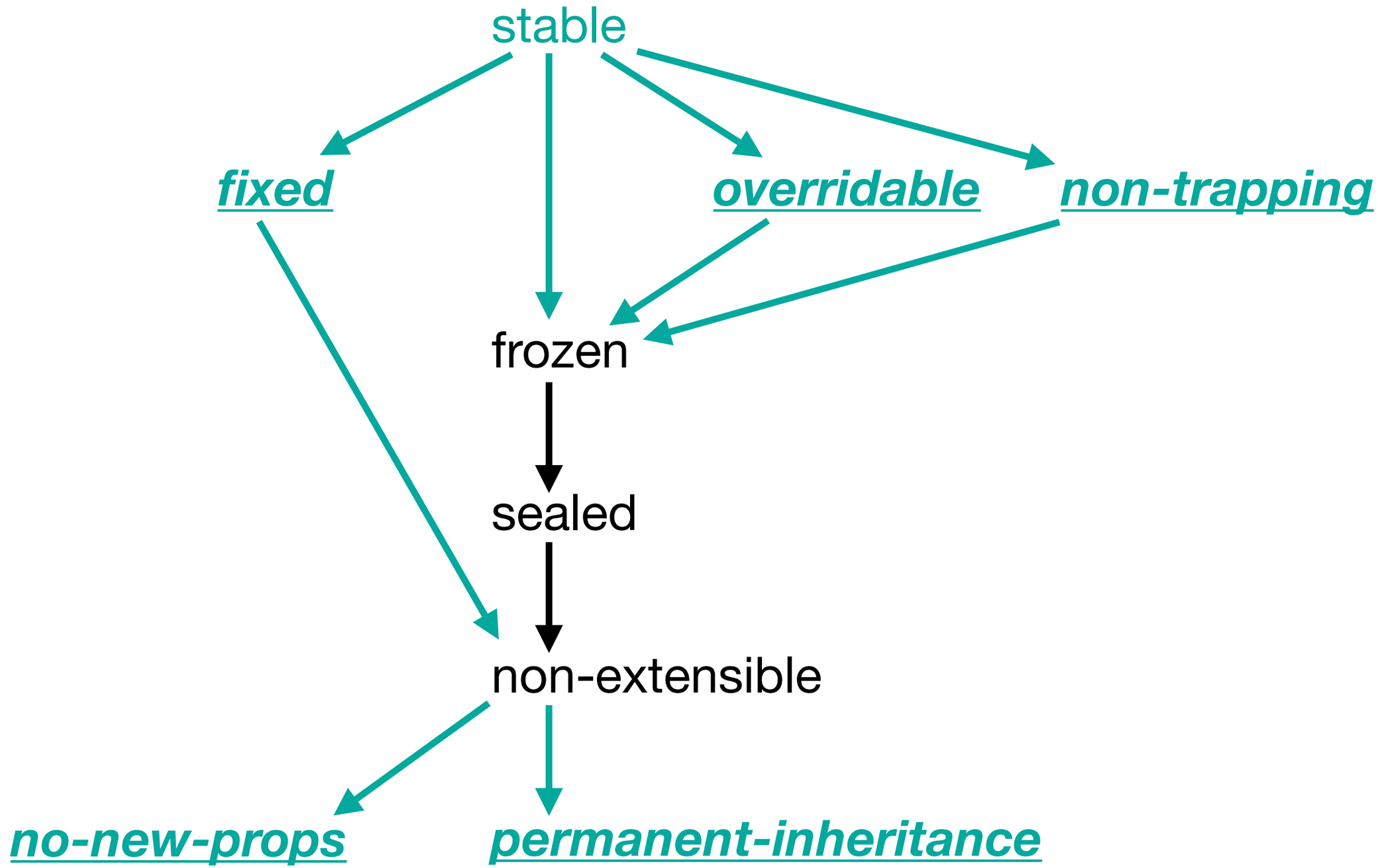


Recap: ***no-new-props***
Rest of non-extensible unbundling

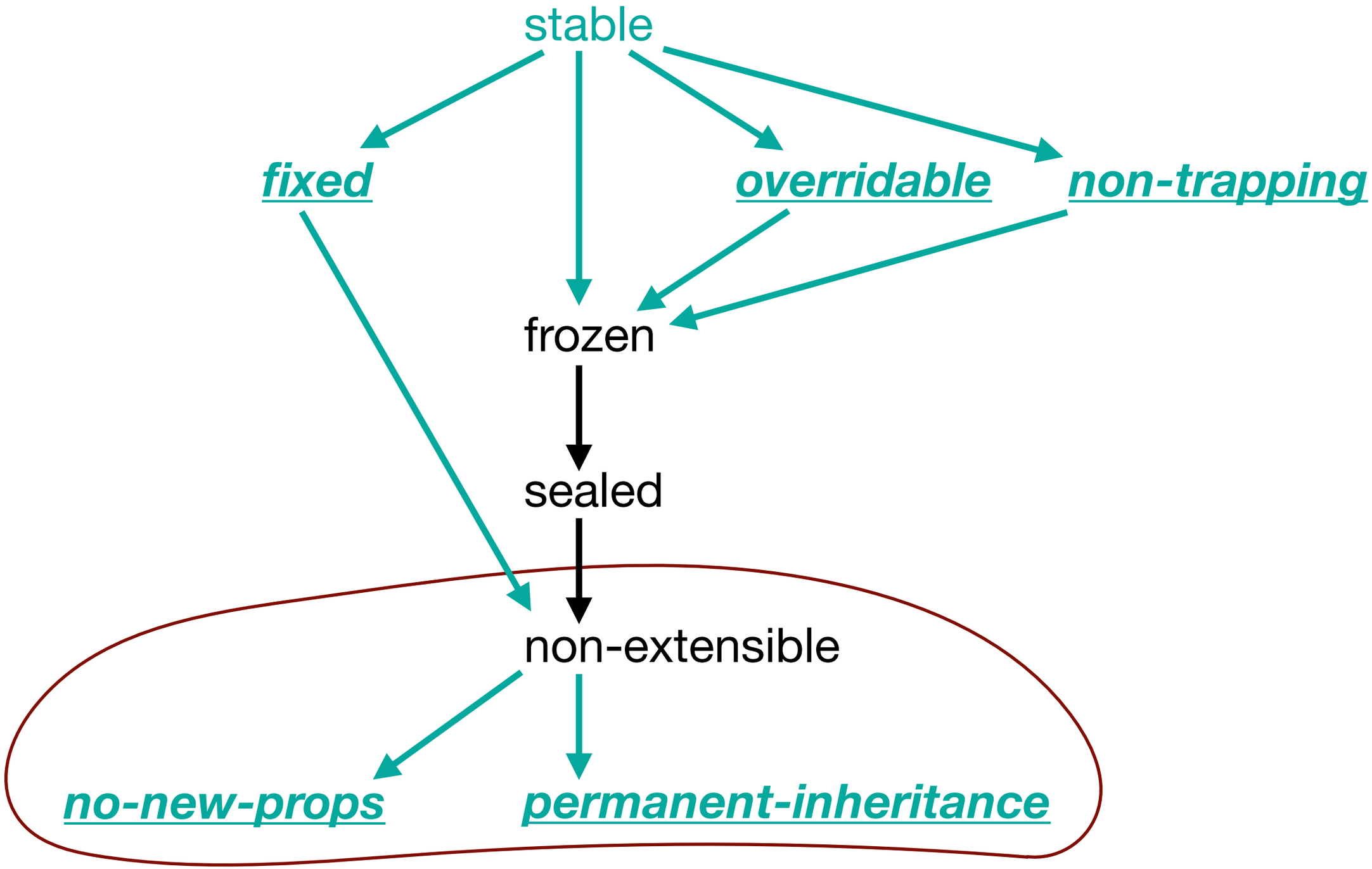



stable
fixed
overridable
non-trapping
frozen
sealed
non-extensible
no-new-props
permanent-inheritance

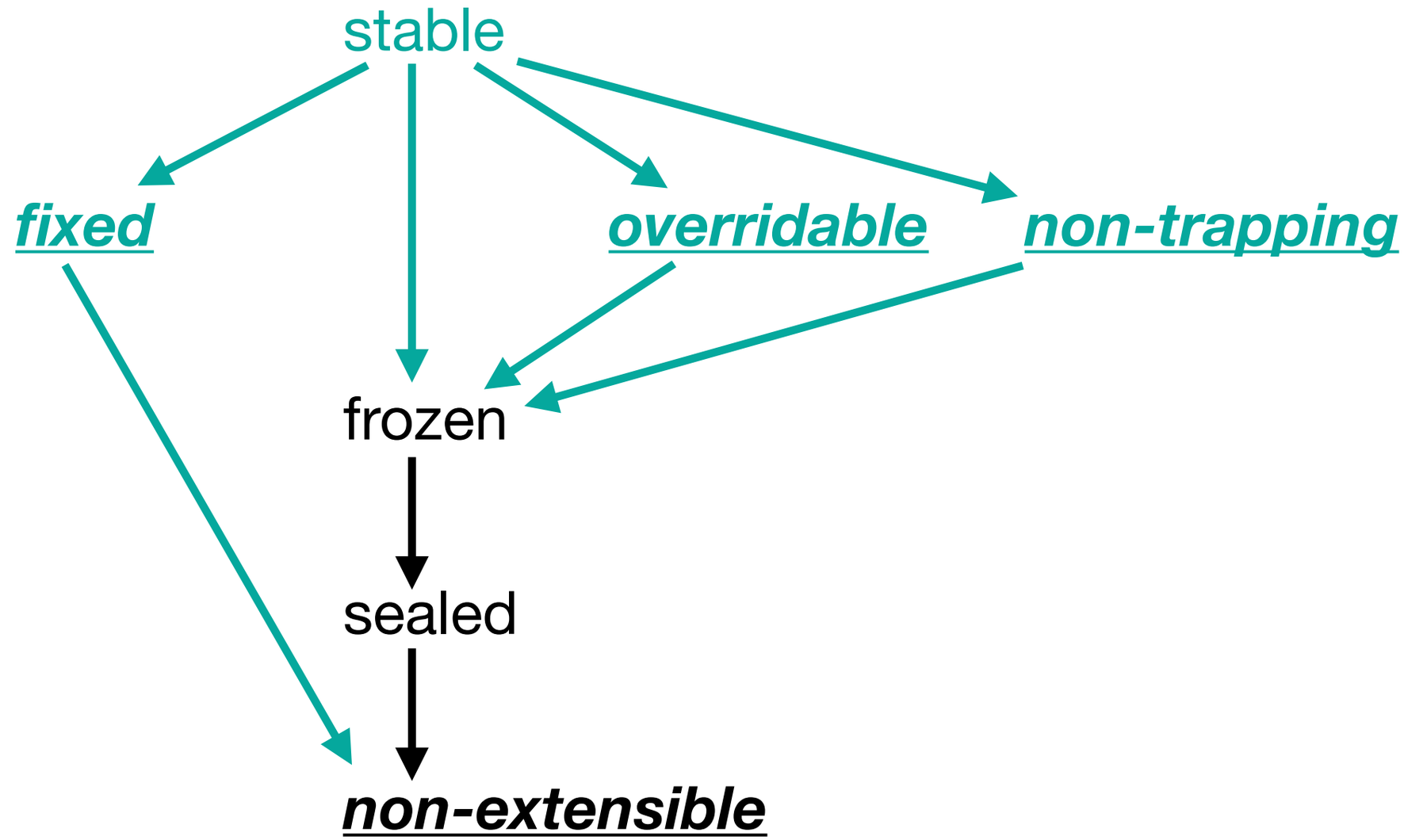
Hoped-for Integrity Traits



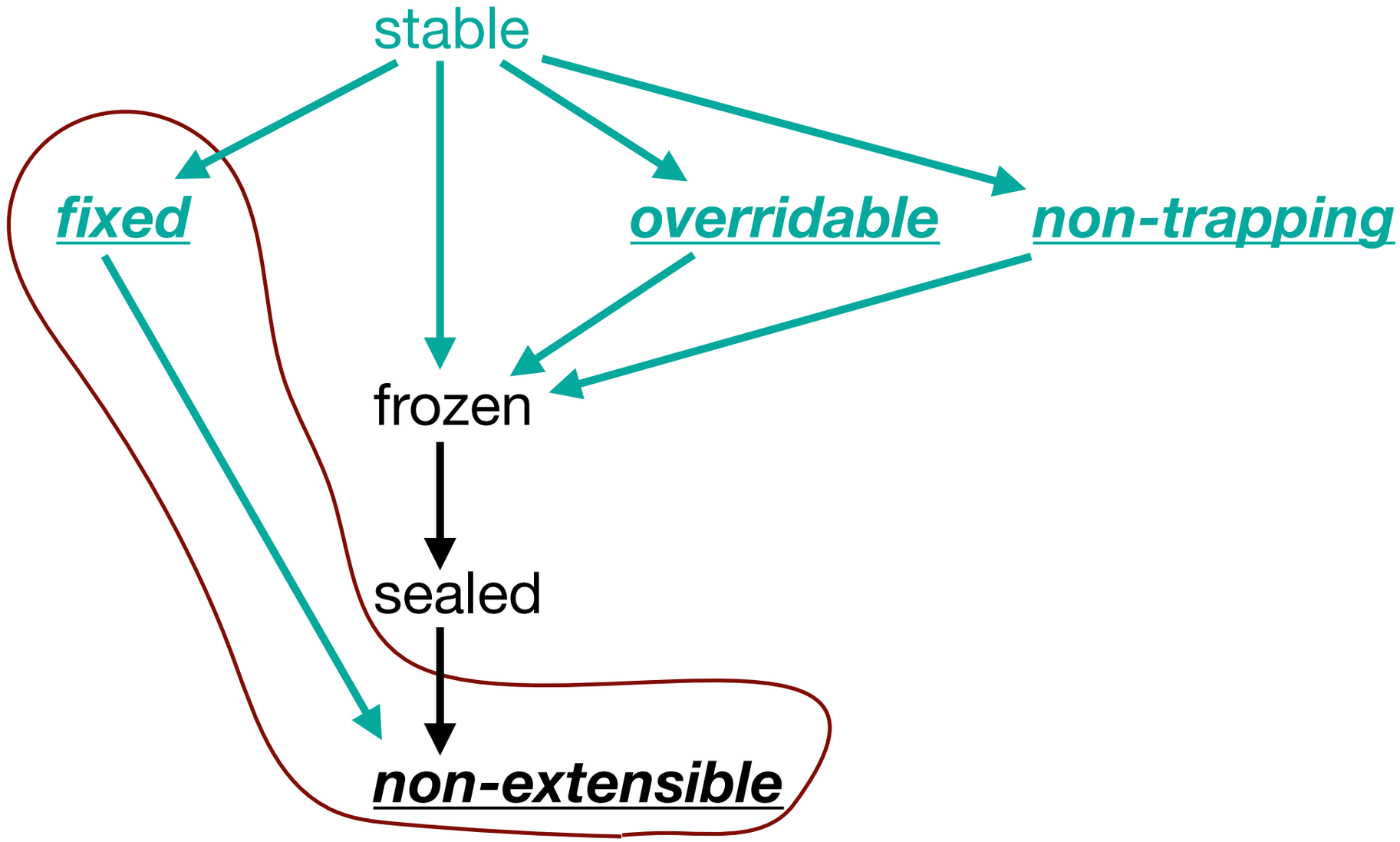
Hoped-for Integrity Traits



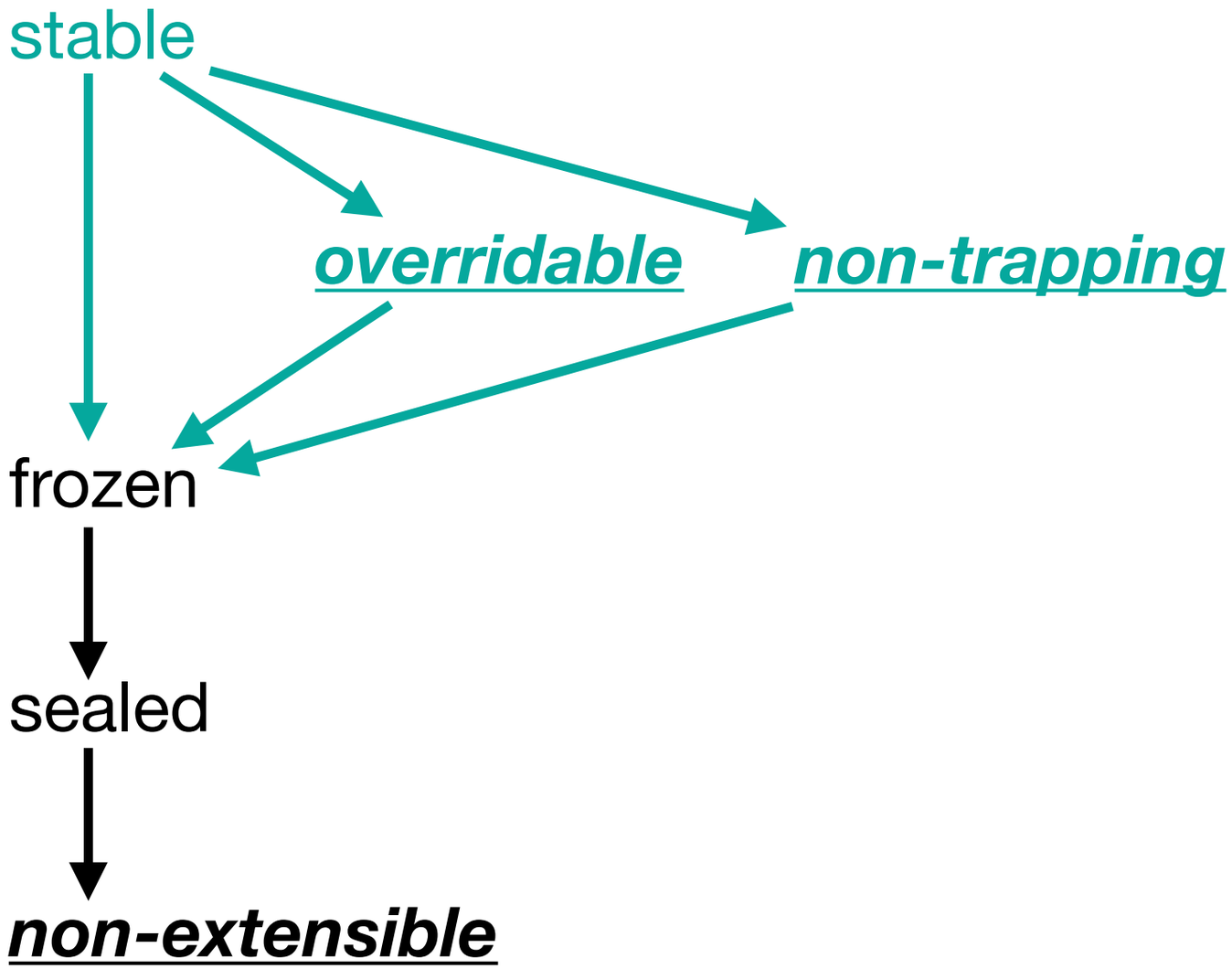
Hoped-for Integrity Traits



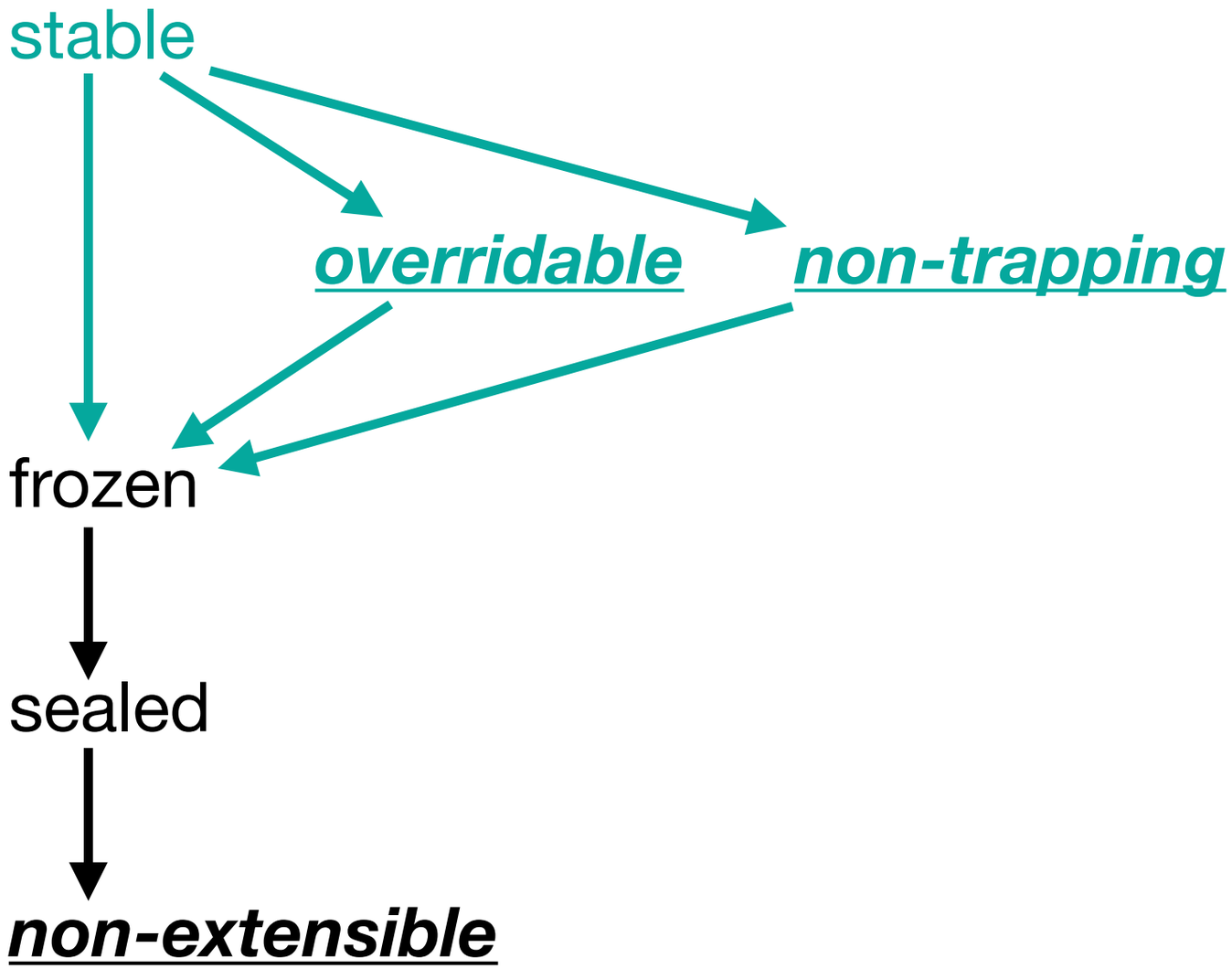
Hoped-for Integrity Traits



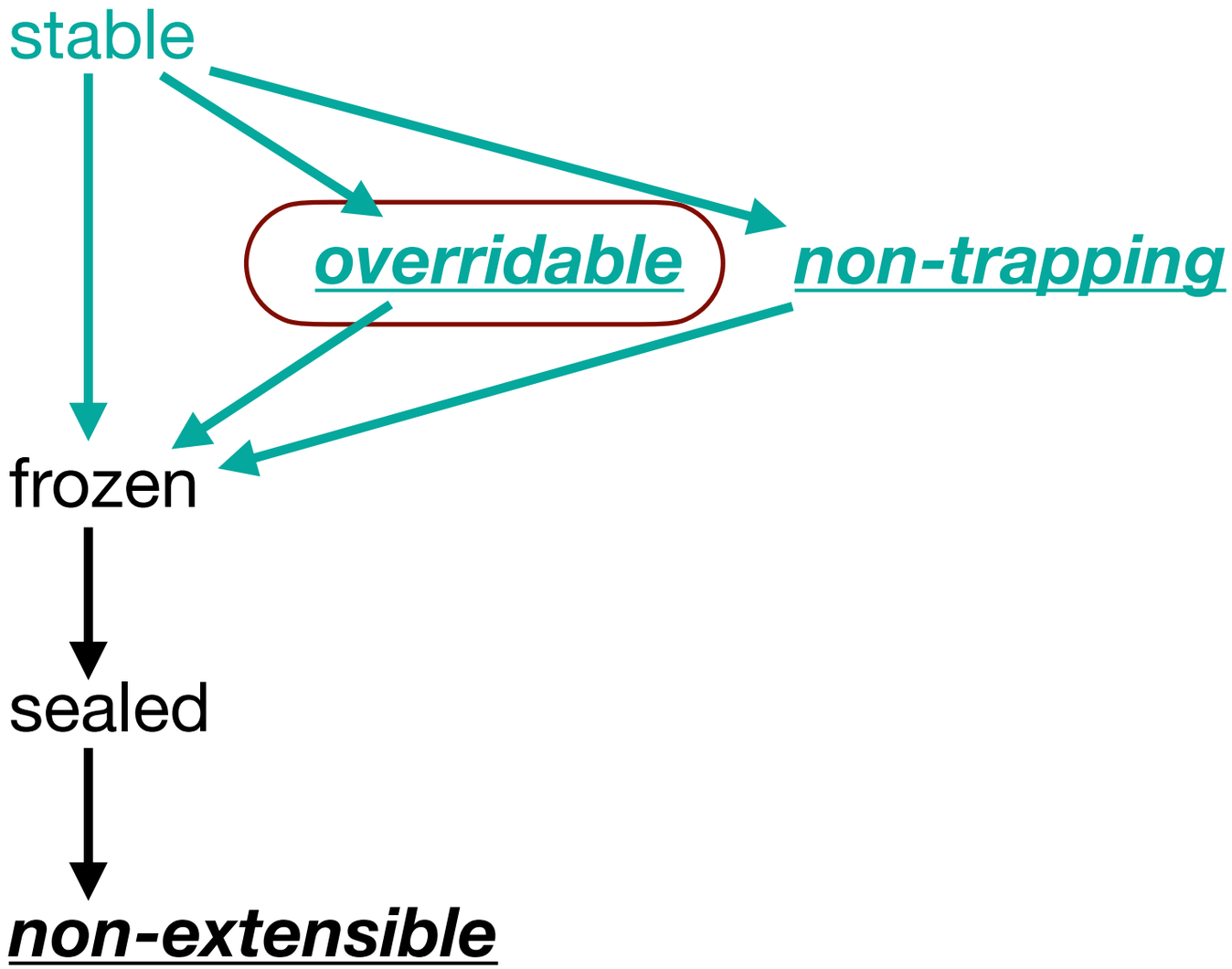
Hoped-for Integrity Traits



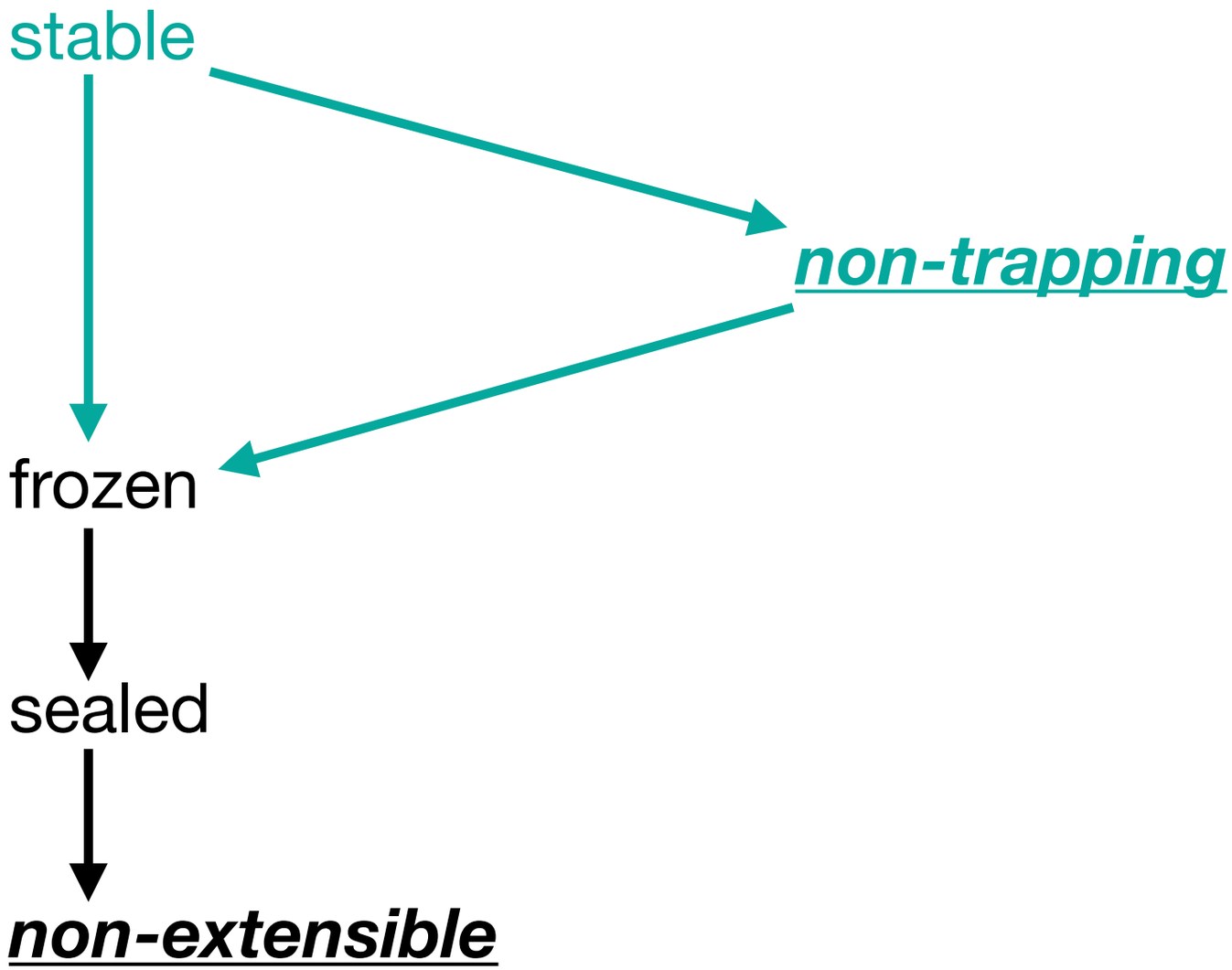
Hoped-for Integrity Traits



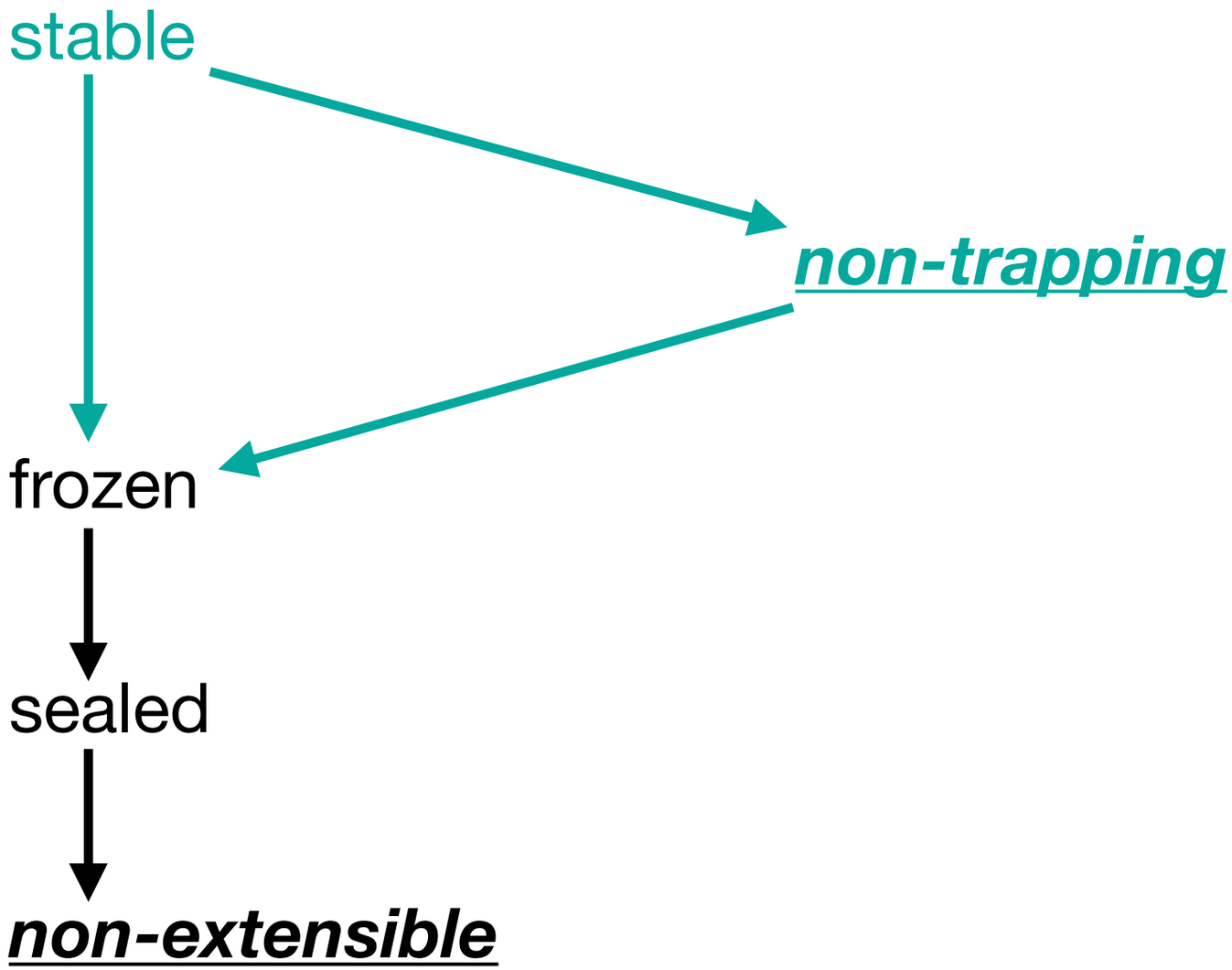
Hoped-for Integrity Traits



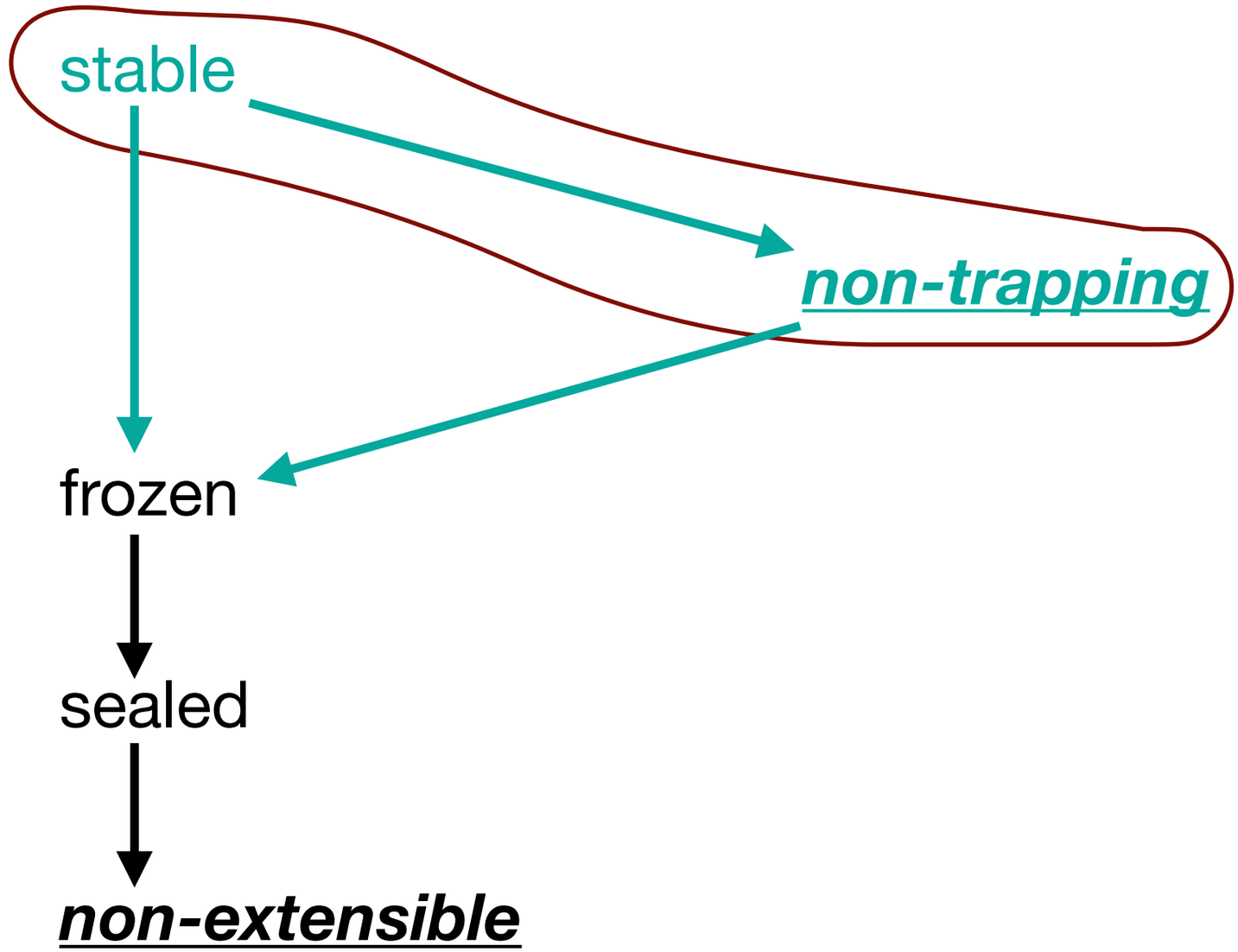
Hoped-for Integrity Traits



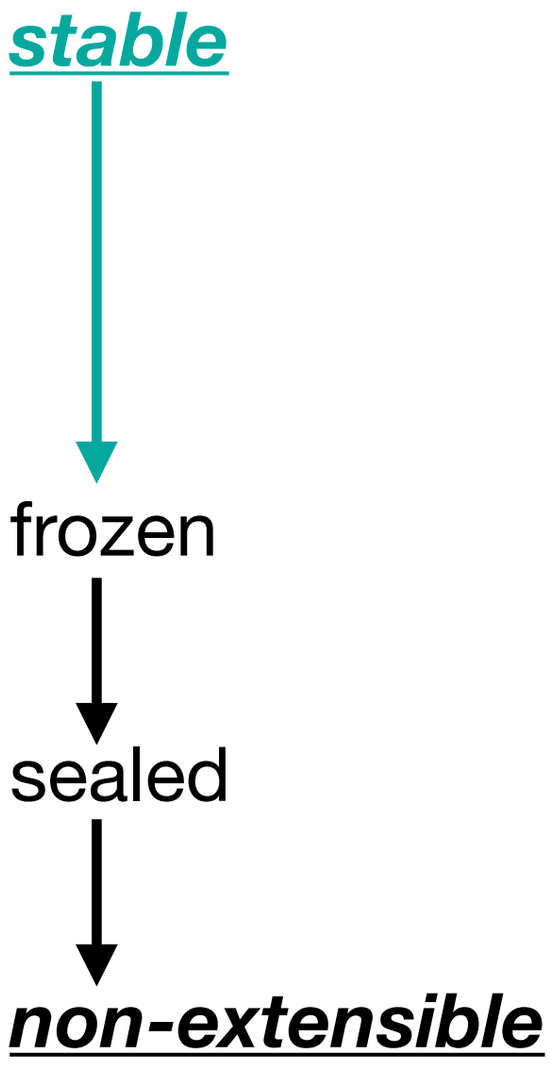
Hoped-for Integrity Traits



Hoped-for Integrity Traits



Hoped-for Integrity Traits



Proxies are Awesome!

Brendan Eich
(w/ Mark Miller & Tom Van Cutsem)

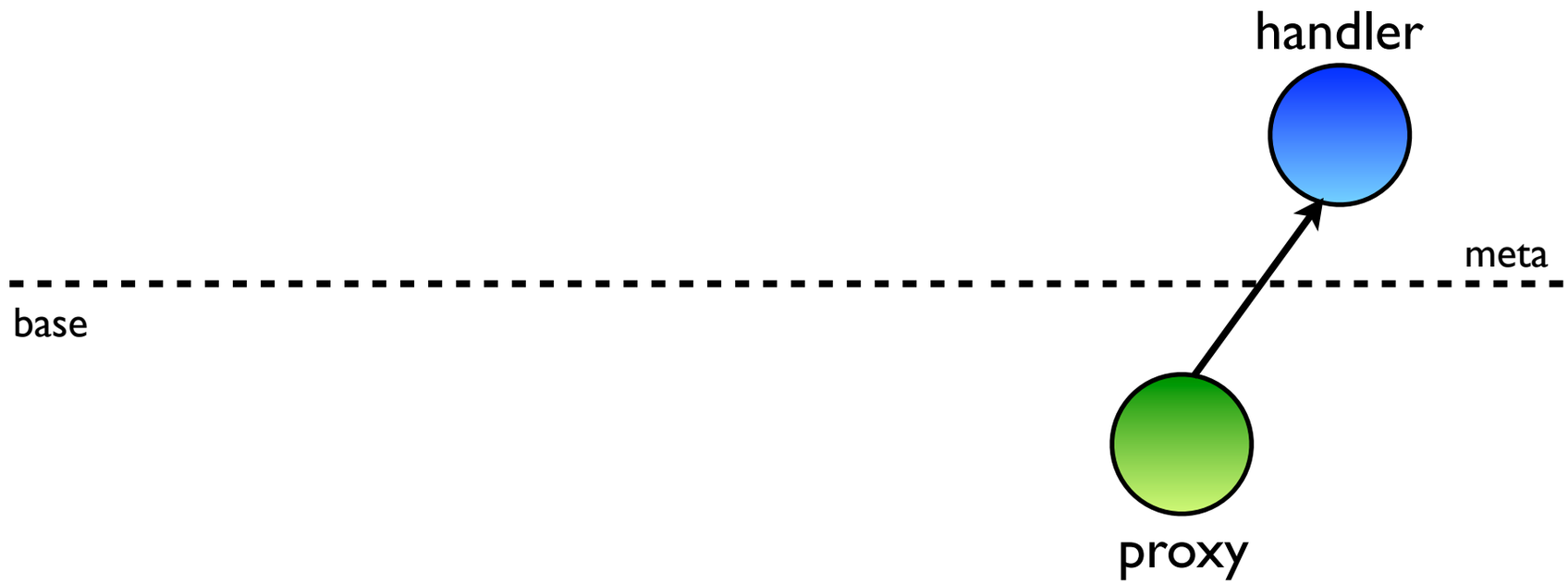
Proxies are Awesome!

Brendan Eich
(w/ Mark Miller & Tom Van Cutsem)

Tuesday, September 28, 2010

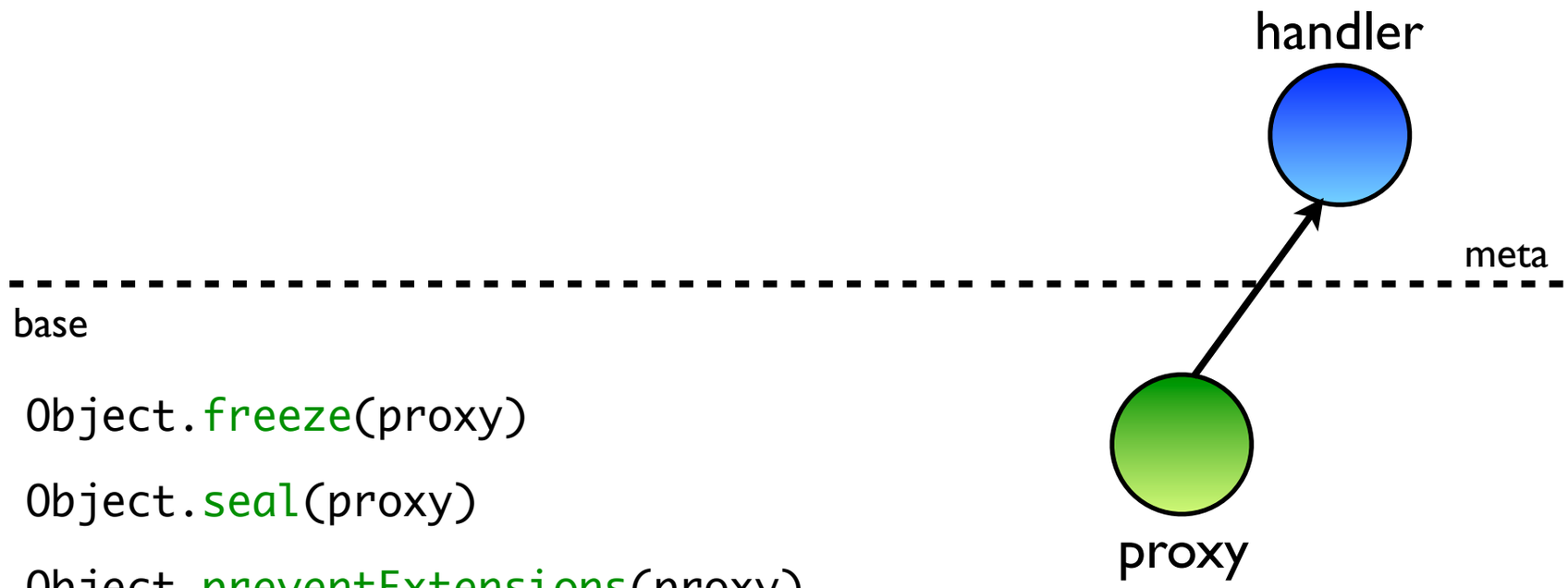
Fixing a Proxy

```
var proxy = Proxy.create(handler, proto);
```



Fixing a Proxy

```
var proxy = Proxy.create(handler, proto);
```

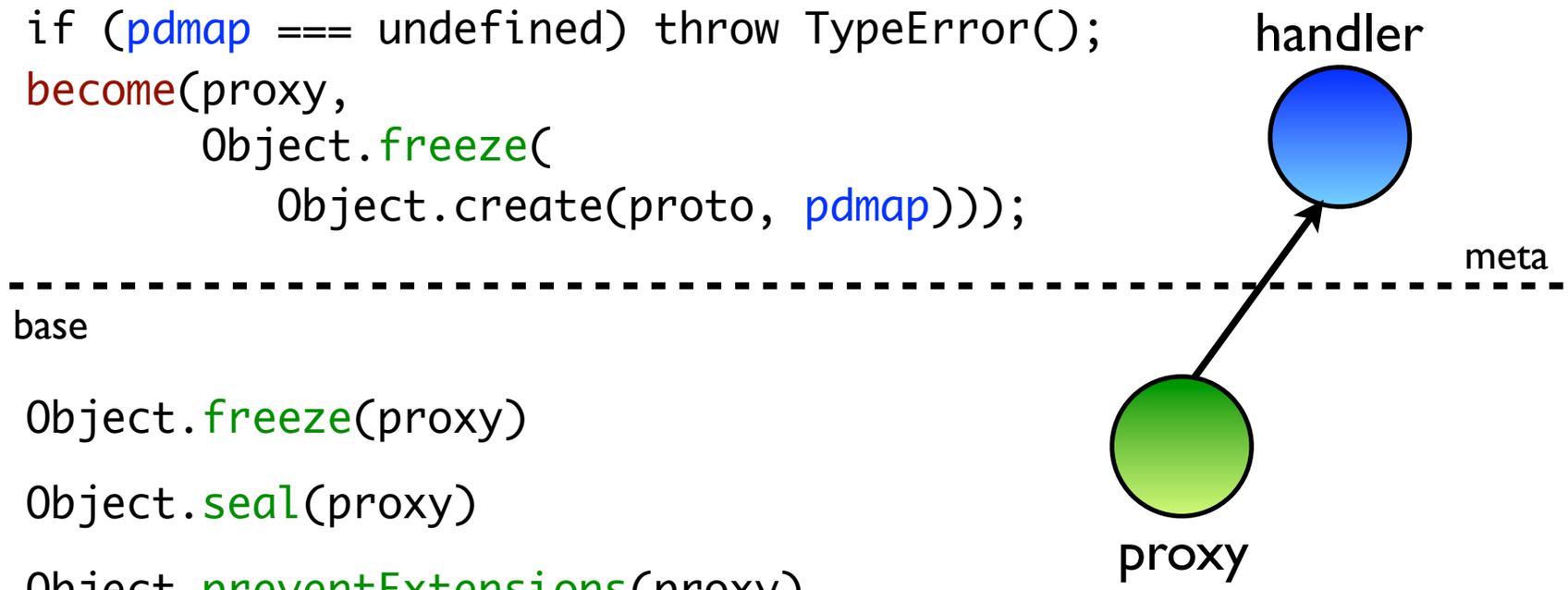



```
Object.freeze(proxy)
Object.seal(proxy)
Object.preventExtensions(proxy)
```

Fixing a Proxy

```
var proxy = Proxy.create(handler, proto);
```

```
var pdmap = handler.fix();
if (pdmap === undefined) throw TypeError();
become(proxy,
       Object.freeze(
          Object.create(proto, pdmap)));
```

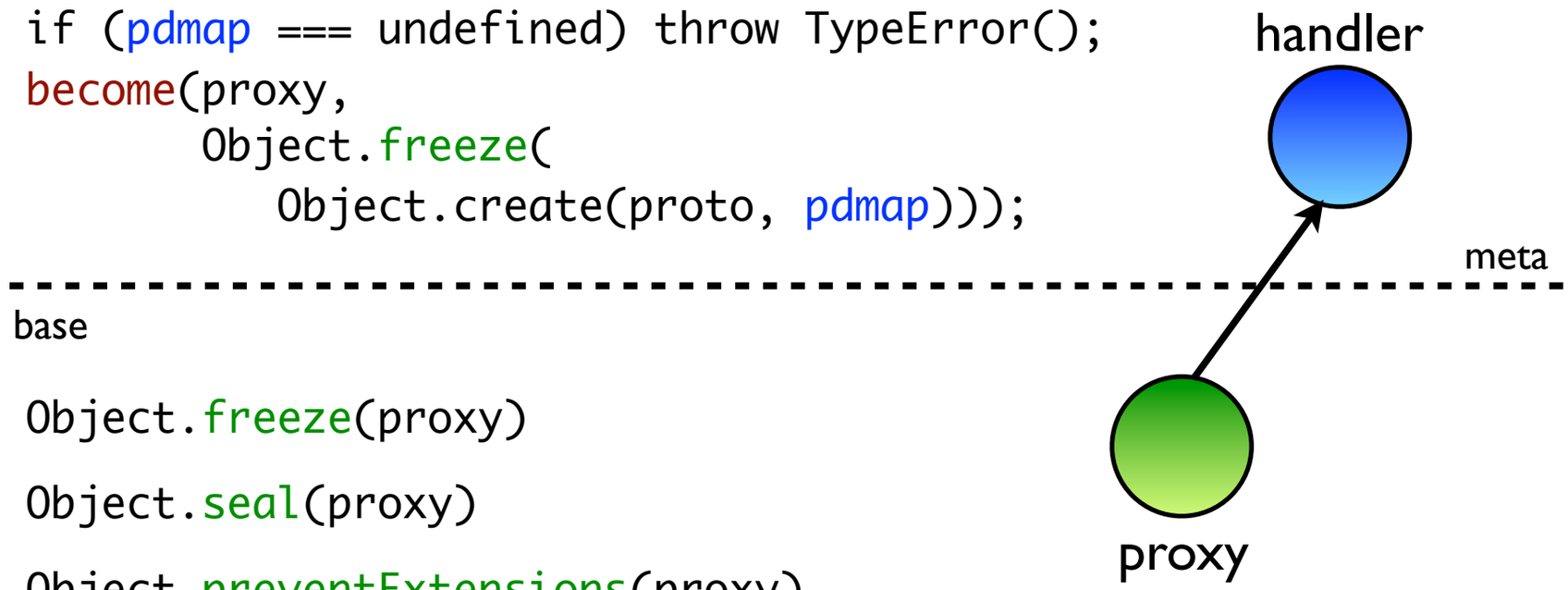



```
Object.freeze(proxy)
Object.seal(proxy)
Object.preventExtensions(proxy)
```

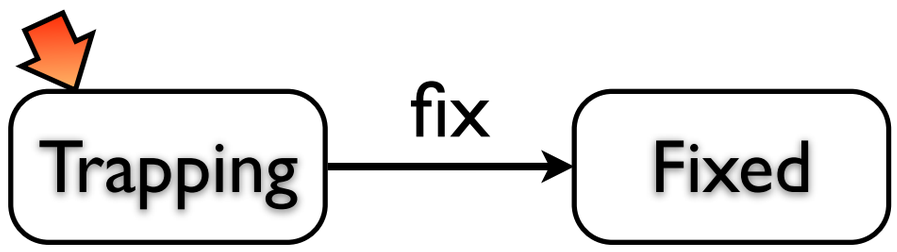
Fixing a Proxy

```
var proxy = Proxy.create(handler, proto);
```

```
var pdmap = handler.fix();
if (pdmap === undefined) throw TypeError();
become(proxy,
       Object.freeze(
          Object.create(proto, pdmap)));
```




```
Object.freeze(proxy)
Object.seal(proxy)
Object.preventExtensions(proxy)
```



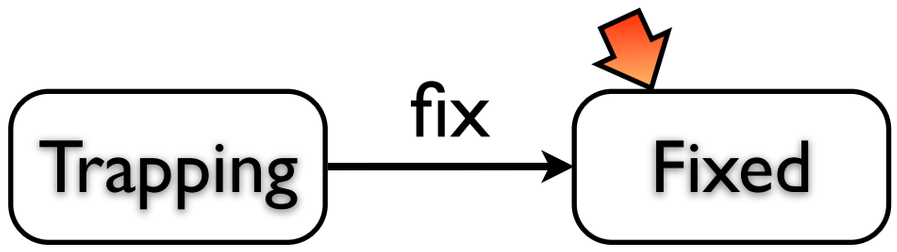
Fixing a Proxy

```
var proxy = Proxy.create(handler, proto);


var pdmap = handler.fix();
if (pdmap === undefined) throw TypeError();
become(proxy,
       Object.freeze(
          Object.create(proto, pdmap)));
```

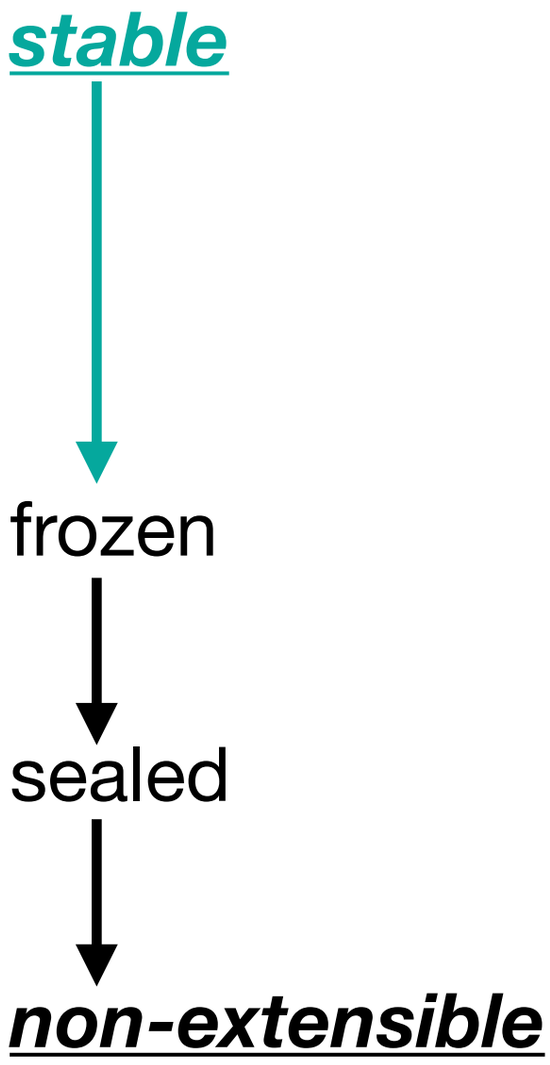
meta

base

```
Object.freeze(proxy)

Object.seal(proxy)

Object.preventExtensions(proxy)
```



Hoped-for Integrity Traits



Hoped-for Integrity Traits