OCapN & Agoric

Layers and Orders

OCapN mtg — Maybe August 8, 2023 ?

(new stuff starting at slide 14)

What Does Fred Want?

```
const optionDesc = harden({
  handle: handle1,
  instance: instanceHandle1,
  installation: coveredCallInstallation,
  description: 'exerciseOption',
  underlyingAssets: { Underlying: moola(3) },
  strikePrice: { StrikePrice: simoleans(7) },
  expirationDate: fooTime(100),
});
```

What Does Fred Want?

```
const optionDesc = harden({
  handle: handle1,
  instance: instanceHandle1,
  installation: coveredCallInstallation,
  description: 'exerciseOption',
  underlyingAssets: { Underlying: moola(3) },
  strikePrice: { StrikePrice: simoleans(7) },
  expirationDate: fooTime(100),
});
```

What Does Fred Want?

```
const optionDesc = harden({
  handle: M.any(),
  instance: M.any(),
  installation: coveredCallInstallation,
  description: 'exerciseOption',
  underlyingAssets: { Underlying: M.gte(moola(3)) },
  strikePrice: { StrikePrice: M.lte(simoleans(7)) },
  expirationDate:
    M.and(M.gte(fooTime(100)), M.lte(fooTime(150))),
});
```

What Does Fred Want?

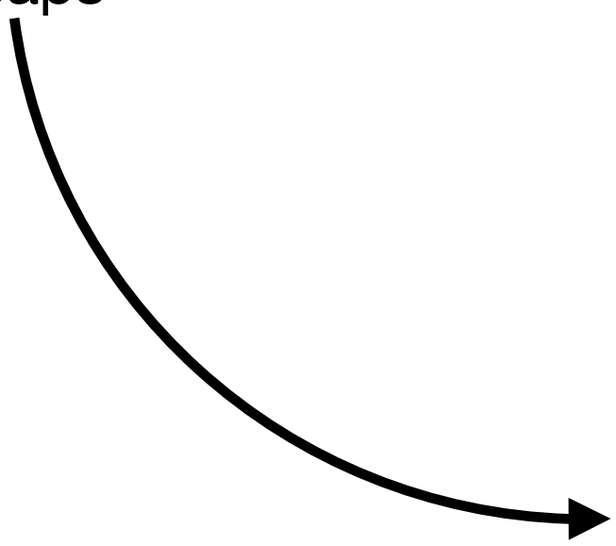
```
const optionDesc = harden({
  handle: M.any(),
  instance: M.any(),
  installation: coveredCallInstallation,
  description: 'exerciseOption',
  underlyingAssets: { Underlying: M.gte(inviteDescSet) },
  strikePrice: { StrikePrice: M.lte(simoleans(7)) },
  expirationDate:
    M.and(M.gte(fooTime(100)), M.lte(fooTime(150))),
});
```

pass-style (passables taxonomy)

`passStyleOf(p: Passable)`

OCapN core data types

marshal (serialization)
`makeMarshal(s2v,v2s)`
smallcaps



pass-style (passables taxonomy)
`passStyleOf(p: Passable)`
OCapN core data types

captp (clists, messages)
`E(bob).foo(carol)`
intervat message passing

marshal (serialization)
`makeMarshal(s2v,v2s)`
smallcaps

pass-style (passables taxonomy)
`passStyleOf(p: Passable)`
OCapN core data types

captp (clists, messages)
`E(bob).foo(carol)`
intervat message passing

marshal (serialization)
`makeMarshal(s2v,v2s)`
smallcaps

patterns (keys, patterns)
`kindOf(p), M`
`compareKeys(x,y)`
sets, bags, maps, matchers

pass-style (passables taxonomy)
`passStyleOf(p: Passable)`
OCapN core data types

captp (clists, messages)
`E(bob).foo(carol)`
intervat message passing

marshal (serialization)
`makeMarshal(s2v,v2s)`
smallcaps

patterns (keys, patterns)
`kindOf(p), M`
`compareKeys(x,y)`
sets, bags, maps, matchers

marshal (rank order)
`compareRank(x,y)`
full order with ties

pass-style (passables taxonomy)
`passStyleOf(p: Passable)`
OCapN core data types

captp (clists, messages)
`E(bob).foo(carol)`
intervat message passing

marshal (serialization)
`makeMarshal(s2v,v2s)`
smallcaps, encodePassable

patterns (keys, patterns)
`kindOf(p), M`
`compareKeys(x,y)`
sets, bags, maps, matchers

marshal (rank order)
`compareRank(x,y)`
full order with ties

pass-style (passables taxonomy)
`passStyleOf(p: Passable)`
OCapN core data types

captp (clists, messages)
`E(bob).foo(carol)`
intervat message passing

exo (defensible remotables)
`makeExoClass(…)`
designed for virtualization

patterns (keys, patterns)
`kindOf(p), M`
`compareKeys(x,y)`
sets, bags, maps, matchers

marshal (serialization)
`makeMarshal(s2v,v2s)`
smallcaps, encodePassable

marshal (rank order)
`compareRank(x,y)`
full order with ties

pass-style (passables taxonomy)
`passStyleOf(p: Passable)`
OCapN core data types

captp (clists, messages)
`E(bob).foo(carol)`
intervat message passing

exo (defensible remotables)
`makeExoClass(…)`
designed for virtualization

store (mutable tables)
`makeMapStore(…)`
designed for virtualization

patterns (keys, patterns)
`kindOf(p), M`
`compareKeys(x,y)`
sets, bags, maps, matchers

marshal (serialization)
`makeMarshal(s2v,v2s)`
smallcaps, encodePassable

marshal (rank order)
`compareRank(x,y)`
full order with ties

pass-style (passables taxonomy)
`passStyleOf(p: Passable)`
OCapN core data types

liveslots (swingset's libc)
connects & virtualizes

captp (clists, messages)
`E(bob).foo(carol)`
intervat message passing

exo (defensible remotables)
`makeExoClass(…)`
designed for virtualization

store (mutable tables)
`makeMapStore(…)`
designed for virtualization

patterns (keys, patterns)
`kindOf(p), M`
`compareKeys(x,y)`
sets, bags, maps, matchers

marshal (serialization)
`makeMarshal(s2v,v2s)`
smallcaps, encodePassable

marshal (rank order)
`compareRank(x,y)`
full order with ties

pass-style (passables taxonomy)
`passStyleOf(p: Passable)`
OCapN core data types

liveslots (swingset's libc)
connects & virtualizes

captp (clists, messages)
`E(bob).foo(carol)`
intervat message passing

exo (defensible remotables)
`makeExoClass(…)`
designed for virtualization

store (mutable tables)
`makeMapStore(…)`
designed for virtualization

patterns (keys, patterns)
`kindOf(p), M`
`compareKeys(x,y)`
sets, bags, maps, matchers

marshal (serialization)
`makeMarshal(s2v,v2s)`
smallcaps, encodePassable

marshal (rank order)
`compareRank(x,y)`
full order with ties

pass-style (passables taxonomy)
`passStyleOf(p: Passable)`
OCapN core data types

	`kindOf` level	`passStyleOf` level	JavaScript level

Classification	`kindOf(p)`
	`M.key()`
	`M.pattern()`
		`passStyleOf(p)`	`typeof j`

Equivalence	`sameKey(k,k)`		`j === j`
			`Object.is(j,j)`
			`sameValueZero(j,j)`

Ordering	`compareKeys(k,k)` `M.gte(k)`	`compareRank(p,p)`	`j <= j` `[...js].sort(compare(j,j))`

Where the parameter

- `j` is for any JavaScript value.
- `p` is for any `Passable`, a subset of JavaScript values.
- `k` is for any `Key`, a subset of passables.

	OCapN name	`passStyleOf`	`typeof`	JS notes
Atoms				
	Null	`null`	`object`	null
	Undefined	`undefined`	`undefined`	
	Boolean	`boolean`	`boolean`	
	Float64	`number`	`number`	Only one `0.0`, Only one `NaN`
	SignedInteger	`bigint`	`bigint`	
	Symbol	`symbol`	`symbol`	well-known & registered only (names TBD)
	String	`string`	`string`	surrogate confusion (TBD)
	ByteString	`byteString` (TBD)	`object`	UInt8Array (TBD)
Containers				
	Sequence	`copyArray`	`object`	Array
	Struct	`copyRecord`	`object`	POJO
	Tagged	`tagged`	`object`	Tagged
Capability				
		`remotable`	`function`	Far function
			`object`	Far object with methods
			`object`	Remote presence
		`promise`	`object`	Promise
Others				
	Error	`error`	`object`	Error

	OCapN name	`passStyleOf`	`typeof`	JS notes
Atoms				
	Null	`null`	`object`	null
	Undefined	`undefined`	`undefined`	
	Boolean	`boolean`	`boolean`	
	Float64	`number`	`number`	Only one `0.0`, Only one `NaN`
	SignedInteger	`bigint`	`bigint`	
	Symbol	`symbol`	`symbol`	well-known & registered only (names TBD)
	String	`string`	`string`	surrogate confusion (TBD)
	ByteString	`byteString` (TBD)	`object`	UInt8Array (TBD)
Containers				
	Sequence	`copyArray`	`object`	Array
	Struct	`copyRecord`	`object`	POJO
	Tagged	`tagged`	`object`	Tagged
Capability				
		`remotable`	`function`	Far function
			`object`	Far object with methods
			`object`	Remote presence
		`promise`	`object`	Promise
Others				
	Error	`error`	`object`	Error

Data

	`kindOf(p)`	`passStyleOf(p)`	meaning
Atoms			
	...	...	primitive data
Containers			
	`copyArray`	`copyArray`	sequence of passable values
	`copyRecord`	`copyRecord`	set of name,passable pairs
	`copySet`	tagged	Set of unique keys
	`copyBag`		Mutiset of key,count pairs
	`copyMap`		Map from keys to passables
Matchers	`match:*`		Non-literal patterns
Guards (TBD)	`guard:*`		Non-pattern guards
Just Tagged	undefined		Not understood to have a kind
Capability			
	`remotable`	`remotable`	behavior + identity
	`promise`	`promise`	Promise
Other			
	`error`	`error`	best efforts loose diagnostics

	kindOf(p)	passStyleOf(p)	meaning
Atoms			
	...	...	primitive data
Containers			
	copyArray	copyArray	sequence of passable values
	copyRecord	copyRecord	set of name,passable pairs
	copySet		Set of unique keys
	copyBag		Mutiset of key,count pairs
	copyMap		Map from keys to passables
Matchers	match:*		Non-literal patterns
Guards (TBD)	guard:*		Non-pattern guards
Just Tagged	undefined		Not understood to have a kind
Capability			
	remotable	remotable	behavior + identity
	promise	promise	Promise
Other			
	error	error	best efforts loose diagnostics

Keys

tagged

	kindOf(p)	passStyleOf(p)	meaning
Atoms			
	...	...	primitive data
Containers			
	copyArray	copyArray	sequence of passable values
	copyRecord	copyRecord	set of name,passable pairs
	copySet	tagged	Set of unique keys
	copyBag		Mutiset of key,count pairs
	copyMap		Map from keys to passables
Matchers	match:*		Non-literal patterns
Guards (TBD)	guard:*		Non-pattern guards
Just Tagged	undefined		Not understood to have a kind
Capability			
	remotable	remotable	behavior + identity
	promise	promise	Promise
Other			
	error	error	best efforts loose diagnostics

Keys

Patterns

	`kindOf(p)`	`compareRank(p,p)`	`compareKeys(k,k)`
Atoms			
	`null`	js	js
	`undefined`	js	js
	`boolean`	js	js
	`number`	num rank< `NaN`	num incomp `NaN`
	`bigint`	js	js
	`symbol`	by name	by name
	`string`	lex (CESU-8?)	lex (CESU-8?)
	`byteString`	lex	lex
Collections			
	`copyArray`	lex	lex
	`copyRecord`	lex keys, value	Pareto
	`copySet`	lex Tagged	subset
	`copyBag`		subbag
	`copyMap`		Pareto
Matchers	`match:*`		N/A
Guards (TBD)	`guard:*`		N/A
Just Tagged	undefined		N/A
Capability			
	`remotable`	same rank	same or incomp
	`promise`	same rank	N/A
Other			
	`error`	same rank	N/A

	`kindOf(p)`	`compareRank(p,p)`	`compareKeys(k,k)`
Atoms			
	...		
Collections			
	`copyArray`	lex	lex
	`copyRecord`	lex keys, value	Pareto
	`copySet`	lex Tagged	subset
	`copyBag`		subbag
	`copyMap`		Pareto
Matchers	`match:*`		N/A
Guards (TBD)	`guard:*`		N/A
Just Tagged	undefined		N/A
Capability			
	`remotable`	same rank	same or incomp
	`promise`	same rank	N/A
Other			
	`error`	same rank	N/A

	`kindOf(p)`	`compareRank(p,p)`	`compareKeys(k,k)`
Atoms			
	...		
Collections			
	`copyArray`	lex	lex
	`copyRecord`	lex keys, value	Pareto
	`copySet`	lex Tagged	subset
	`copyBag`		subbag
	`copyMap`		Pareto
Capability			
	`remotable`	same rank	same or incomp

	`kindOf(p)`	`compareRank(p,p)`	`compareKeys(k,k)`
Atoms			
	...		
Collections			
	`copyArray`	lex	lex
	`copyRecord`	lex keys, value	Pareto
	`copySet`	lex Tagged	subset
	`copyBag`		subbag
	`copyMap`		Pareto
Capability			
	`remotable`	same rank	same or incomp

	`kindOf(p)`	`compareRank(p,p)`	`compareKeys(k,k)`
Atoms			
	...		
Collections			
	`copyArray`	lex	lex
	`copyRecord`	lex keys, value	Pareto
	`copySet`	lex Tagged	subset
	`copyBag`		subbag
	`copyMap`		Pareto
Capability			
	`remotable`	same rank	same or incomp

```
M.and(M.gte(makeCopySet(['b','e'])),
      M.lte(makeCopySet(['a','b','c','e','f'])))
```

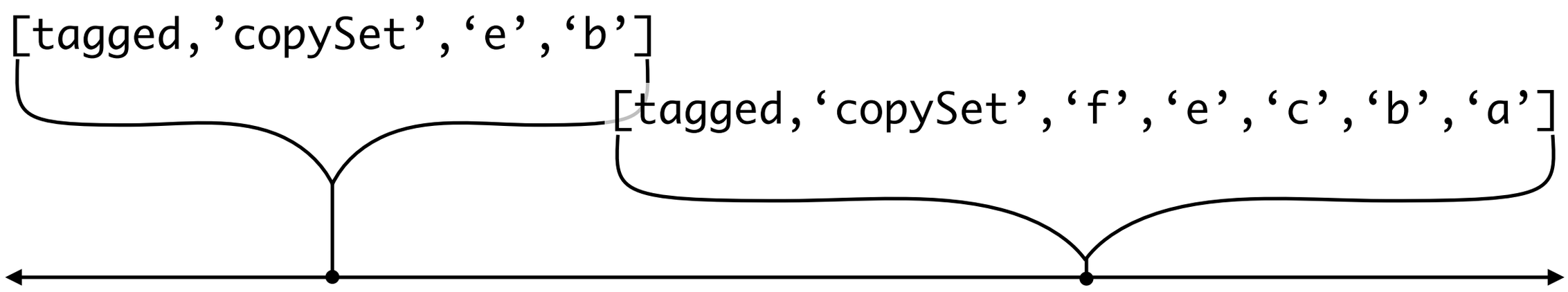
```
M.and(M.gte(makeCopySet(['b','e'])),
      M.lte(makeCopySet(['a','b','c','e','f'])))
```

{'b','e'}..{'a','b','c','e','f'}

```
M.and(M.gte(makeCopySet([‘b’,’e’])),
      M.lte(makeCopySet([‘a’,’b’,’c’,‘e’,‘f’])))

                  {‘b’,‘e’}..{‘a’,’b’,‘c’,‘e’,‘f’}

const makeCopySet = elements =>
  makeTagged(‘copySet’, reverseSort(elements));
```

```
M.and(M.gte(makeCopySet([‘b’,’e’])),
      M.lte(makeCopySet([‘a’,’b’,’c’,‘e’,‘f’])))

                  {‘b’,‘e’}..{‘a’,’b’,‘c’,‘e’,‘f’}

const makeCopySet = elements =>
  makeTagged(‘copySet’, reverseSort(elements));
```

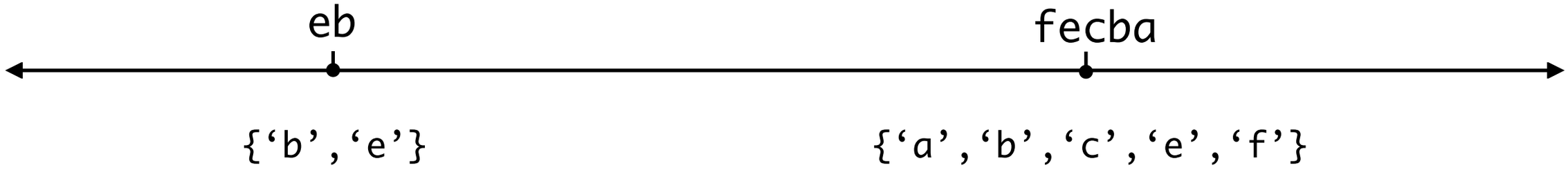
Sets subset



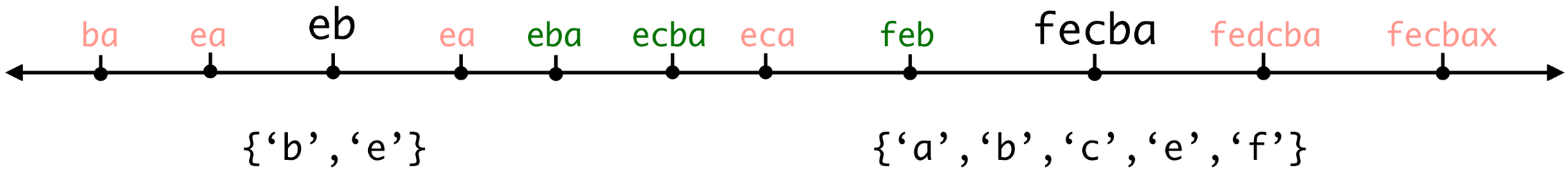
```
M.and(M.gte(makeCopySet([‘b’,’e’])),
      M.lte(makeCopySet([‘a’,’b’,’c’,‘e’,‘f’])))

                  {‘b’,‘e’}..{‘a’,’b’,‘c’,‘e’,‘f’}

const makeCopySet = elements =>
  makeTagged(‘copySet’, reverseSort(elements));
```

Sets subset

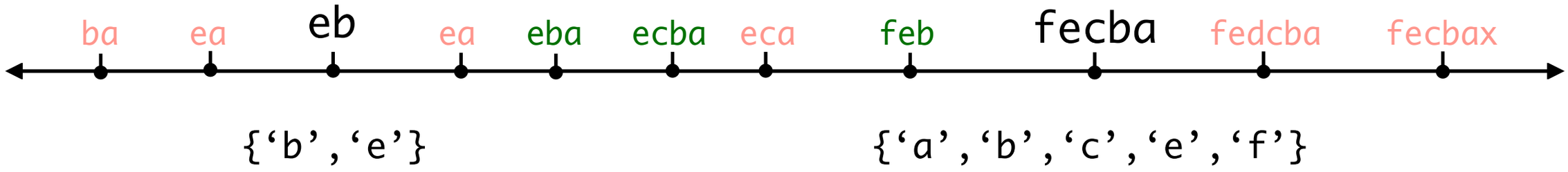
Sets subset



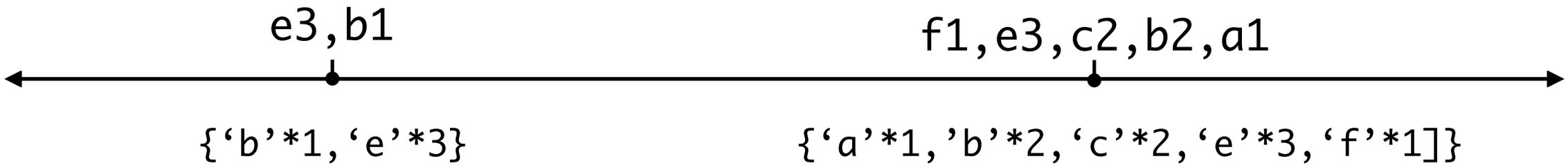
Sets subset



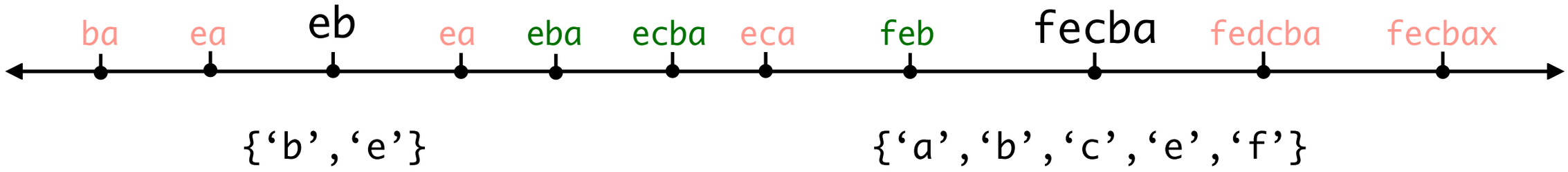
Sets subset



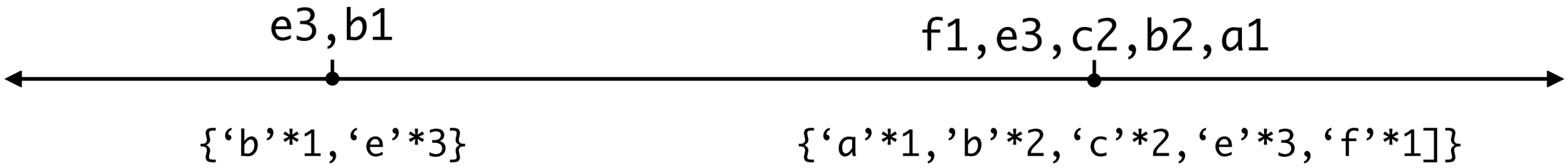
Bags subbag



Sets subset



Bags subbag



Maps Pareto

What Does Fred Want?

```
const optionDesc = harden({
  handle: M.any(),
  instance: M.any(),
  installation: coveredCallInstallation,
  description: 'exerciseOption',
  underlyingAssets: { Underlying: M.gte(inviteDescSet) },
  strikePrice: { StrikePrice: M.lte(simoleans(7)) },
  expirationDate:
    M.and(M.gte(fooTime(100)), M.lte(fooTime(150))),
});
```

liveslots (swingset's libc)
connects & virtualizes

captp (clists, messages)
`E(bob).foo(carol)`
intervat message passing

exo (defensible remotables)
`makeExoClass(…)`
designed for virtualization

store (mutable tables)
`makeMapStore(…)`
designed for virtualization

patterns (keys, patterns)
`kindOf(p), M`
`compareKeys(x,y)`
sets, bags, maps, matchers

marshal (serialization)
`makeMarshal(s2v,v2s)`
smallcaps, encodePassable

marshal (rank order)
`compareRank(x,y)`
full order with ties

pass-style (passables taxonomy)
`passStyleOf(p: Passable)`
OCapN core data types

liveslots (swingset's libc)
connects & virtualizes

captp (clists, messages)
`E(bob).foo(carol)`
intervat message passing

exo (defensible remotables)
`makeExoClass(…)`
designed for virtualization

store (mutable tables)
`makeMapStore(…)`
designed for virtualization

patterns (keys, patterns)
`kindOf(p), M`
`compareKeys(x,y)`
sets, bags, maps, matchers

marshal (serialization)
`makeMarshal(s2v,v2s)`
smallcaps, encodePassable

marshal (rank order)
`compareRank(x,y)`
full order with ties

pass-style (passables taxonomy)
`passStyleOf(p: Passable)`
OCapN core data types

liveslots (swingset's libc)
connects & virtualizes

captp (clists, messages)
`E(bob).foo(carol)`
intervat message passing

exo (defensible remotables)
`makeExoClass(…)`
designed for virtualization

store (mutable tables)
`makeMapStore(…)`
designed for virtualization

patterns (keys, patterns)
`kindOf(p), M`
`compareKeys(x,y)`
sets, bags, maps, matchers

marshal (serialization)
`makeMarshal(s2v,v2s)`
smallcaps, encodePassable

marshal (rank order)
`compareRank(x,y)`
full order with ties

pass-style (passables taxonomy)
`passStyleOf(p: Passable)`
OCapN core data types

liveslots (swingset's libc)
connects & virtualizes

store (mutable tables)
`makeMapStore(…)`
designed for virtualization

exo (defensible remotables)
`makeExoClass(…)`
designed for virtualization

captp (clists, messages)
`E(bob).foo(carol)`
intervat message passing

patterns (keys, patterns)
`kindOf(p), M`
`compareKeys(x,y)`
sets, bags, maps, matchers

marshal (serialization)
`makeMarshal(s2v,v2s)`
smallcaps, encodePassable

marshal (rank order)
`compareRank(x,y)`
full order with ties

pass-style (passables taxonomy)
`passStyleOf(p: Passable)`
OCapN core data types