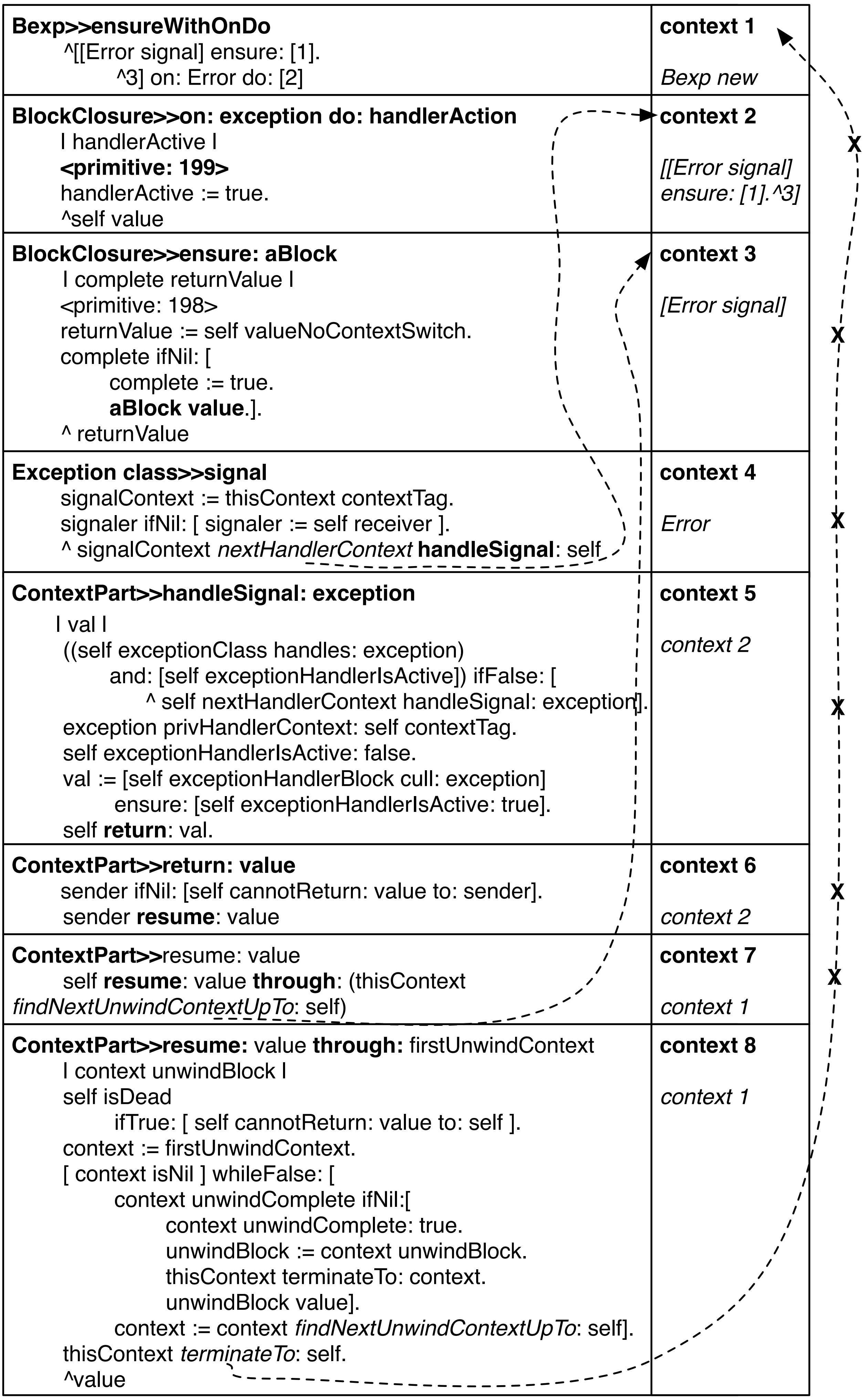

```
Bexp>>ensureWithOnDo
	^[[Error signal] ensure: [1].
		^3] on: Error do: [2]
```

context 1

Bexp new

```
BlockClosure>>on: exception do: handlerAction
	| handlerActive |
	<primitive: 199>
	handlerActive := true.
	^self value
```

context 2

[[Error signal] ensure: [1].^3]

```
BlockClosure>>ensure: aBlock
	| complete returnValue |
	<primitive: 198>
	returnValue := self valueNoContextSwitch.
	complete ifNil: [
		complete := true.
		aBlock value.].
	^ returnValue
```

context 3

[Error signal]

```
Exception class>>signal
	signalContext := thisContext contextTag.
	signaler ifNil: [ signaler := self receiver ].
	^ signalContext nextHandlerContext handleSignal: self
```

context 4

Error

```
ContextPart>>handleSignal: exception
	| val |
	((self exceptionClass handles: exception)
		and: [self exceptionHandlerIsActive]) ifFalse: [
			^ self nextHandlerContext handleSignal: exception].
	exception privHandlerContext: self contextTag.
	self exceptionHandlerIsActive: false.
	val := [self exceptionHandlerBlock cull: exception]
		ensure: [self exceptionHandlerIsActive: true].
	self return: val.
```

context 5

context 2

```
ContextPart>>return: value
	sender ifNil: [self cannotReturn: value to: sender].
	sender resume: value
```

context 6

context 2

```
ContextPart>>resume: value
	self resume: value through: (thisContext
findNextUnwindContextUpTo: self)
```

context 7

context 1

```
ContextPart>>resume: value through: firstUnwindContext
	| context unwindBlock |
	self isDead
		ifTrue: [ self cannotReturn: value to: self ].
	context := firstUnwindContext.
	[ context isNil ] whileFalse: [
		context unwindComplete ifNil:[
			context unwindComplete: true.
			unwindBlock := context unwindBlock.
			thisContext terminateTo: context.
			unwindBlock value].
		context := context findNextUnwindContextUpTo: self].
	thisContext terminateTo: self.
	^value
```

context 8

context 1