

Example of non-tail call, needs to remember (if • ...) context.

(if (f x)
 (return 0)
 (return 1))

⋮

(if • (return 0) (return 1))
 (f x)

This is OK!
 (tail calls)

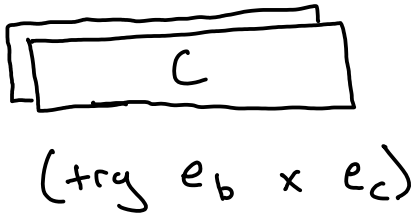
(try e_b x e_c)

(throw e)

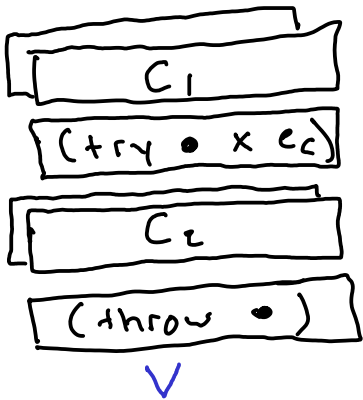
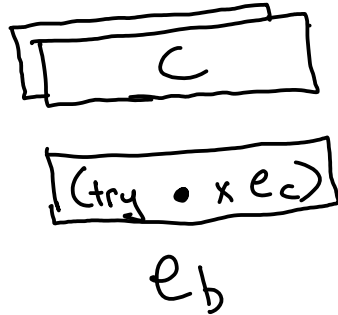
(try (+ 1 (throw "err"))) x ((+ "msg was: " x))

(try (+ 1 2) x x) → 3

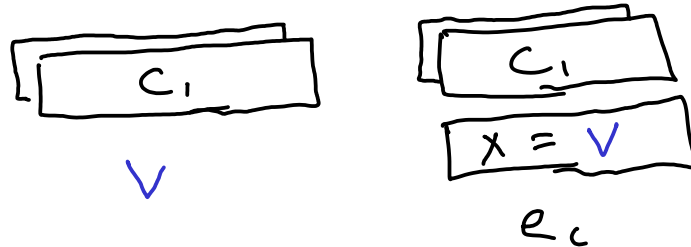
↓
"msg was: err"



⇒



⇒



↑
this is different
behavior from above
(doesn't evaluate catch block)

(gen e)

(next c)

(yield e)

```
def nats(n)
```

```
  (gen
```

```
    i = 0
```

```
    while True
```

```
      (yield i)
```

```
      i = i + 1 )
```

```
def nats():
```

```
  i = 0
```

```
  while True:
```

```
    yield i
```

```
    i = i + 1
```

```
g = nats()
```

```
(next g) ...
```



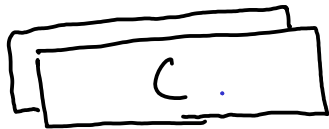
(gen e)

⇒



gen(get_bindings(c), e, false)

→ is-generator-started



(next •)

⇒

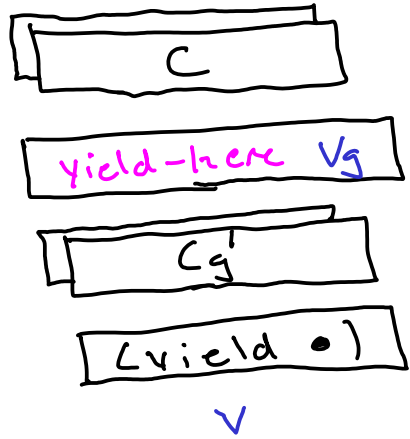


yield-here v_g

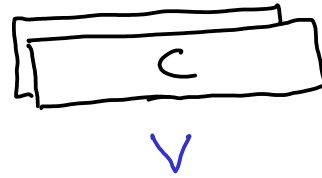


e_g

v_g = gen(c_g, e_g, false)

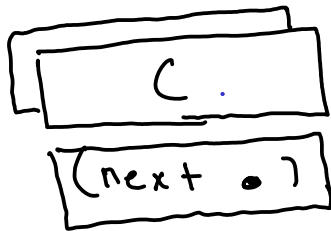


⇒

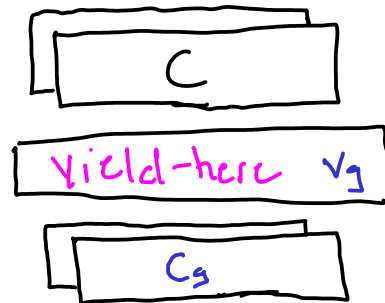


$v_g := \text{gen}(c_g', e, \text{true})$
 ↖ state update on v_g

(v_g before update is $\text{gen}(c_g, e, \{\text{true or false}\})$)



⇒



None

$v_g = \text{gen}(c_g, e_g, \text{true})$

STACK COPYING ON YIELD

