Una expressió qualsevol on apareix `Continuation callcc: [ :k | ... ]` proporciona un entorn per a aquest enviament de missatge #callcc:

La continuació és precisament aquest entorn, que està esperant el retorn de l'enviament de missatge `Continuation callcc: [ :k | ... ]`. Aleshores, com funciona `Continuation callcc: [ :k | ... ]`?

Una cosa que hem de tenir MOLT clara és que sempre que es fa `Continuation callcc: [ :k | ... ]`, s'avaluarà el bloc `[ :k | ... ]`

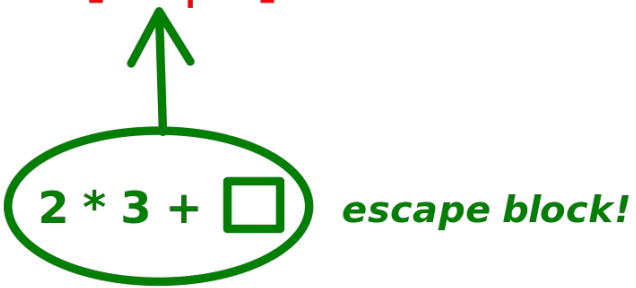
`Continuation callcc: [ :k | ... ]` = `[ :k | ... ] value:` *<la continuació>*

L'entorn del que parlàvem més amunt és l'expressió on apareix `Continuation callcc: [ :k | ... ]`, esperant un valor de retorn d'aquest enviament de missatge en el mateix punt on s'ha fet l'enviament de #callcc:

Ens podem imaginar la continuació com un block especial, d'un paràmetre, on el paràmetre està posat just en el punt on hem fet `Continuation callcc: [ :k | ... ]`. És com si hagués un forat que espera un valor retornat. Aquest block és un block especial perquè és un *escape block*, és a dir, un block que ell mateix es converteix en allò que queda per fer, abandonant qualsevol cosa que s'estés executant.

Exemple:

2 * 3 + (Continuation callcc: [:k | 5])



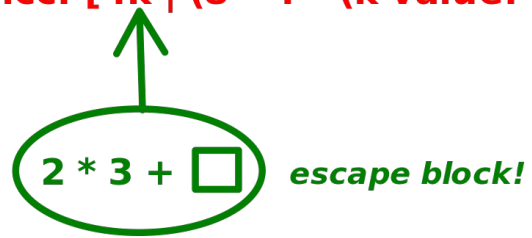
S'avalua el block [:k | 5] amb resultat 5

Fixem-nos que NO fem servir k, no fem servir la continuació

2 * 3 + (Continuation callcc: [:k | 5]) => 2 * 3 + 5 => 11

Exemple:

2 * 3 + (Continuation callcc: [:k | (8 * 4 * (k value: 5)) traceCr])



S'avalua el block [:k | (8 * 4 * (k value: 5)) traceCr]

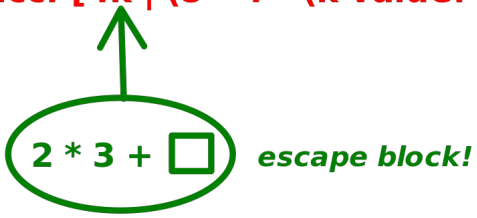
Aquí sí estem utilitzant (l'estem avaluant) la continuació.

Si la continuació NO fos un escape block, si fos un block normal: 8 * 4 * (2 * 3 + 5) => 352 i aquest número s'escriuria al Transcript...

Però això NO passa!

Exemple:

2 * 3 + (Continuation callcc: [:k | (8 * 4 * (k value: 5)) traceCr])



S'avalua el block [:k | (8 * 4 * (k value: 5)) traceCr]

Com que la continuació és com un *escape block*, allò que estava pendent de fer en el moment d'avaluar-la, (32 + ...) traceCr, s'abandona. El que fem és talment *com si tornéssim* a l'expressió principal, 2 * 3 + ... i fessim servir el paràmetre que passem a l'avaluació de la continuació (en aquest cas 5) com a valor retornat per Continuation callcc: [:k | (8 * 4 * (k value: 5)) traceCr]

Així doncs, 2 * 3 + (Continuation callcc: [:k | ...]) => 2 * 3 + 5 => 11 i no s'escriu RES al Transcript

Exemple:

Al tanto! ***escape block!***

```
x := □.
x ifFalse: [ cont value: true ].
x
```



```
| x cont |
x := Continuation callcc: [ :k | cont := k. false ].
x ifFalse: [ cont value: true ].
x
```

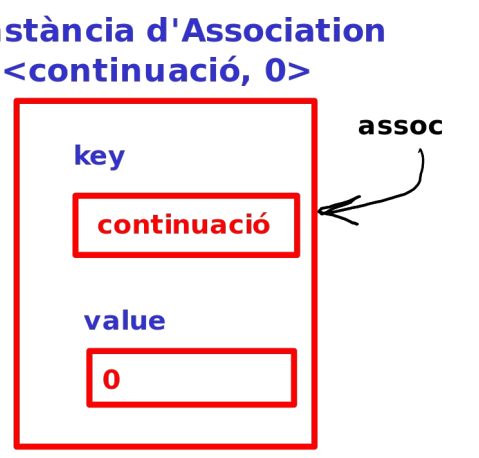
```
x := true.
x ifFalse: [ cont value: true ].
x
```

i tot el que quedava pendent s'abandona

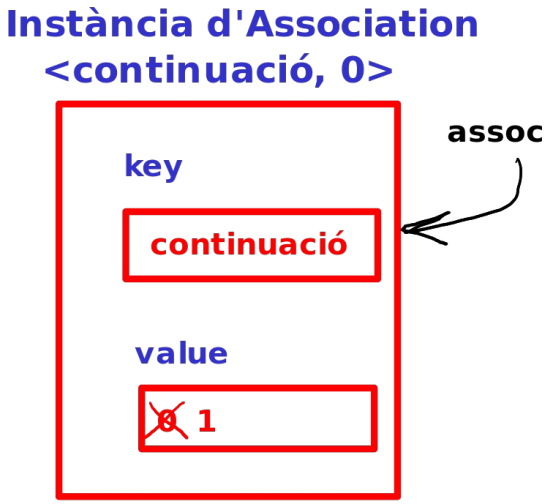
Exemple:

```
| assoc |
assoc := Continuation callcc: [ :cc | cc -> 0 ].
assoc value: assoc value + 1.
self assert: assoc value ~= 5.
assoc value = 4
        ifFalse: [ assoc key value: assoc ]
```

El bloc retorna la instància d'Association

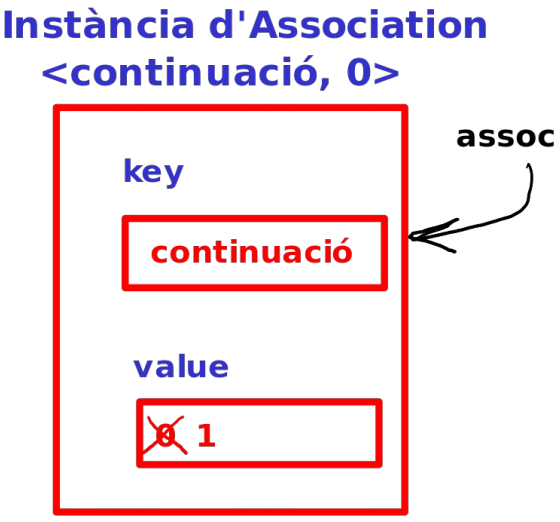


```
| assoc |
assoc := Continuation callcc: [ :cc | cc -> 0 ].
assoc value: assoc value + 1.
self assert: assoc value ~= 5.
assoc value = 4
        ifFalse: [ assoc key value: assoc ]
```



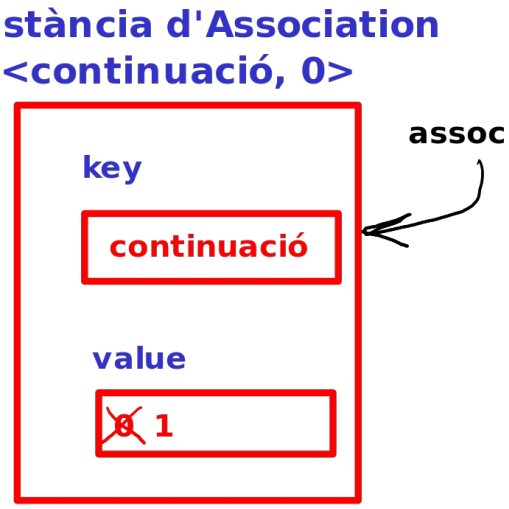
```
| assoc |
assoc := Continuation callcc: [ :cc | cc -> 0 ].
assoc value: assoc value + 1.
self assert: assoc value ~= 5.
assoc value = 4
        ifFalse: [ assoc key value: assoc ]
```

assoc value val 1, així que això és fals



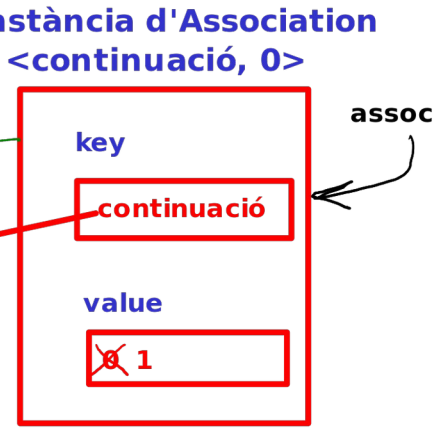
```
| assoc |
assoc := Continuation callcc: [ :cc | cc -> 0 ].
assoc value: assoc value + 1.
self assert: assoc value ~= 5.
assoc value = 4
        ifFalse: [ assoc key value: assoc ]
```

(assoc key) és la continuació, i li envio el missatge #value: amb paràmetre l'objecte referenciat per assoc



Avaluar la continuació amb paràmetre assoc vol dir que tot el que s'estava fent s'abandona, i posem en el seu lloc la continuació...

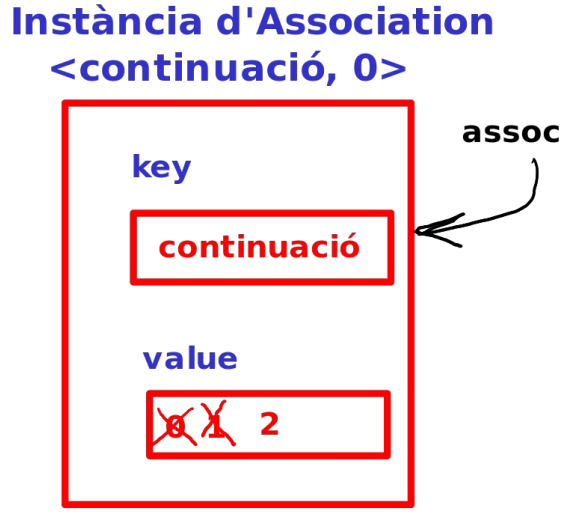
```
| assoc |
assoc := [].
assoc value: assoc value + 1.
self assert: assoc value ~= 5.
assoc value = 4
        ifFalse: [ assoc key value: assoc ]
```



... amb assoc com a paràmetre, que vol dir que és com si l'enviament del missatge #callcc: hagués retornat assoc

I ara continuem executant, tornant a incrementar l'atribut value d'assoc...

```
| assoc |
assoc := ...
assoc value: assoc value + 1.
self assert: assoc value ~= 5.
assoc value = 4
        ifFalse: [ assoc key value: assoc ]
```



I així vaig fent, fins que l'atribut value d'assoc valgui 4, i el programa s'acaba (hauria de ser obvi per quina raó s'acaba, us ho deixo com exercici)

Fixeu-vos que l'atribut key d'assoc, la continuació, no canvia MAI, i que les assignacions a assoc en avaluar la continuació consisteixen essencialment en assignar (la referència a) assoc, a assoc, és a dir, es queda amb el mateix valor que ja tenia.

```
| assoc |
assoc := Continuation callcc: [ :cc | cc -> 0 ].
assoc value: assoc value + 1.
self assert: assoc value ~= 5.
assoc value = 4
        ifFalse: [ assoc key value: assoc ]
```

Aquest és un exemple de com iterar amb continuacions. Més endavant en veurem de més sofisticats