$v_1 = \lambda\_:() \to \text{else}\ (\text{send}\ ()\ w)$

$s_2 = \text{let}\ (x, a) = \text{receive}\ a\ \text{in}\ \text{Wait}\ a$

$s = \text{let}\ (w, a) = \text{new}\ !().?().\text{end}!\ \text{in}\ s_2; (\text{fork}\ v_1); x$

Type Derivations

$$\frac{\dagger\text{-Const}}{\vdash \text{send} : () \to (!().\text{end}!) \to \text{end}!} \qquad \frac{}{() : ()}\ \dagger\text{-Const}$$

$$\frac{\vdash \text{send} : () \to (!().\text{end}!) \to \text{end}! \qquad () : ()}{\vdash \text{send}\ () : (!().\text{end}!) \to \text{end}!}\ \dagger\text{-APP}$$

$$\frac{}{w : !().\text{end}! \vdash w : !().\text{end}!}\ \dagger\text{-VAR}$$

$$\frac{\vdash \text{send}\ () : (!().\text{end}!) \to \text{end}! \qquad w : !().\text{end}! \vdash w : !().\text{end}!}{w : !().\text{end}! \vdash \text{send}\ ()\ w : \text{end}!}\ \dagger\text{-APP}$$

$$\frac{\dagger\text{-Const}}{\vdash \text{else} : \text{end}! \to ()} \qquad \frac{w : !().\text{end}! \vdash \text{send}\ ()\ w : \text{end}!}{w : !().\text{end}! \vdash \text{else}\ (\text{send}\ ()\ w) : ()}\ \dagger\text{-APP}$$

$$\frac{w : !().\text{end}! \vdash \text{else}\ (\text{send}\ ()\ w) : ()}{** \; w : !().\text{end}! \vdash \lambda\_:() \to \text{else}\ (\text{send}\ ()\ w) : () \to ()}\ \dagger\text{-ABS}$$

$$\frac{}{x : () \vdash x : ()}\ \dagger\text{-Var} \qquad \frac{}{\vdash \text{fork} : (() \to ()) \to ()}\ \dagger\text{-Const} \qquad **$$

$$\frac{\vdash (\lambda\_:().x) : () \to () \qquad w : !().\text{end}! \vdash \text{fork}\ (\lambda\_:() \to \text{else}\ (\text{send}\ ()\ w)) : ()}{w : !().\text{end}! \vdash (\lambda\_:().x)\ \text{fork}\ (\lambda\_:() \to \text{else}\ (\text{send}\ ()\ w)) : ()}\ \dagger\text{-APP}$$

$$\frac{w : !().\text{end}! \vdash (\lambda\_:().x)\ \text{fork}\ (\lambda\_:() \to \text{else}\ (\text{send}\ ()\ w)) : ()}{w : !().\text{end}! \vdash (\lambda\_:().(\lambda\_:().x)\ \text{fork}\ (\lambda\_:() \to \text{else}\ (\text{send}\ ()\ w))) : () \to ()}\ \dagger\text{-ABS}$$

$$\frac{}{* \; a : ?().\text{end}? \vdash a : ?().\text{end}?}\ \dagger\text{-VAR}$$

$$\frac{\dagger\text{-CONST}}{\vdash \text{receive} : ?().\text{end}? \to () \times \text{end}?} \qquad * \qquad \frac{\dagger\text{-Const}}{\vdash \text{Wait} : \text{end}? \to ()} \qquad \frac{\dagger\text{-VAR}}{a : \text{end}? \vdash a : \text{end}?}$$

$$\frac{\vdash \text{receive}\ a : () \times \text{end}? \qquad x : () , a : \text{end}? \vdash \text{Wait}\ a : ()}{a : ?().\text{end}? \vdash \text{receive}\ a : () \times \text{end}? \qquad x : () , a : \text{end}? \vdash \text{Wait}\ a : ()}\ \dagger\text{-APP}$$

$$\frac{a : ?().\text{end}? \vdash \text{receive}\ a : () \times \text{end}? \qquad x : () , a : \text{end}? \vdash \text{Wait}\ a : ()}{a : ?().\text{end}? \vdash \text{let}\ (x, a) = \text{receive}\ a\ \text{in}\ \text{Wait}\ a : ()}\ \dagger\text{-LET}$$

$$\frac{a : ?().\text{end}? \vdash \text{let}\ (x, a) = \text{receive}\ a\ \text{in}\ \text{Wait}\ a : ()}{w : (!().\text{end}!) , a : (?().\text{end}?) \vdash (\lambda\_:().(\lambda\_:().x)\ \text{fork}\ (\lambda\_:() \to \text{else}\ (\text{send}\ ()\ w)))\ \text{let}\ (x, a) = \text{receive}\ \text{in}\ \text{Wait}\ a : ()}\ \dagger\text{-APP}$$

$$\frac{w : (!().\text{end}!) , a : (?().\text{end}?) \vdash (\lambda\_:().(\lambda\_:().x)\ \text{fork}\ (\lambda\_:() \to \text{else}\ (\text{send}\ ()\ w)))\ \text{let}\ (x, a) = \text{receive}\ \text{in}\ \text{Wait}\ a : () \qquad w : (!().\text{end}!) , a : (?().\text{end}?) \vdash \text{let}\ (x, a) = \text{receive}\ a\ \text{in}\ \text{Wait}\ a ; \text{fork}\ (\lambda\_:() \to \text{else}\ (\text{send}\ ()\ w)) ; x : ()}{\vdash \text{let}\ (w, a) = \text{new}\ !().\text{end}!\ \text{in}\ s_2 ; (\text{fork}\ v_1) ; x : ()}\ \dagger\text{-LET}$$

$$\frac{\dagger\text{-NEW}}{\vdash \text{new}\ !\text{unit}.\text{end}! : (!().\text{end}!) \times (?().\text{end}?)} \qquad w : (!().\text{end}!) , a : (?().\text{end}?) \vdash s_2 ; (\text{fork}\ v_1) ; x : ()$$

Reduction Derivation Steps

<e> → (νxy)<let (w,r) = new !unit.end! in e2;(fork v1);x (R-NEW)

(e2) (e1)

let (w,r) = (x,y) in e2;(fork v1);x → let(x,y) = receive y in wait y;fork(λ_:()→ close (send () x));x (E-SPLIT)

<let (w,r) = (x,y) in e2;(fork v1);x> → <e2;(fork v1);x> (R-THREAD)

(νxy) <let(w,r) = (x,y) in e2;(fork v1); x → (νxy)<e2;(fork v1);x> (R-RES)

<(λ_:().(λ_:().x)e2)(fork v1)> → <(λ_:().(λ_:().x)e2)()> | <v1()> (R-FORK)

(νxy)<(λ_:().(λ_:().x)e2)(fork v1)> → (νxy) <(λ_:().(λ_:().x)e2)()>|<v1()> (R-RES)

⇕

e2;(fork v1);x

(λ_:().(λ_:().x)e2)() → (λ_:().x)e2 (E-APP)

<(λ_:().(λ_:().x)e2)()> → <(λ_:().x)e2> (R-THREAD)

<(λ_:().(λ_:().x)e2)()>|<v1()> → <(λ_:().x)e2>|<v1()> (R-PAR)

(νxy) <(λ_:().(λ_:().x) e2)()> | <v1()> → (νxy)<(λ_:().x)e2>|<v1()> (R-RES)

Repeating the last derivation tree, we have that:

(νxy)<(λ_:().x) e2>|<v1()> → (νxy)<x>|<v1()>

We can conclude that, since the process <x> can never end up in a "wait x" or "close y" format.

With this, we can conclude that R-Close will never apply and we don't have to reduce any further to show this.