Application du résultat de bad1

Soit l'application suivante

```
> (add2 1)
((lambda(x)
    (if (null? '(add1 add1))
        x
        (add1 ((bad1 add1) x))))
 1)
```

Choisir une réduction parmi :

- Application de la lambda
- Calcul du if
- Appel récursif
- Forme normale (pas de réduction possible)

Application du résultat de bad1 (1)

```
( (lambda(x)
    (if (null? '(add1 add1))
        x
        (add1 ((bad1 add1) x))))
1)
```

$\xrightarrow{\beta}$

```
(if (null? '(add1 add1))
    1
    (add1 ((bad1 add1) 1)))
```

Étape suivante : λ ou if ou récursion ou forme normale

Application du résultat de bad1 (2)

```
(if (null? '(add1 add1))
    1
    (add1 ((bad1 add1) 1)))
```

$\xrightarrow{\beta}$

```
(add1 ((bad1 add1) 1))
```

Étape suivante : λ ou if ou récursion ou forme normale

Application du résultat de bad1 (3)

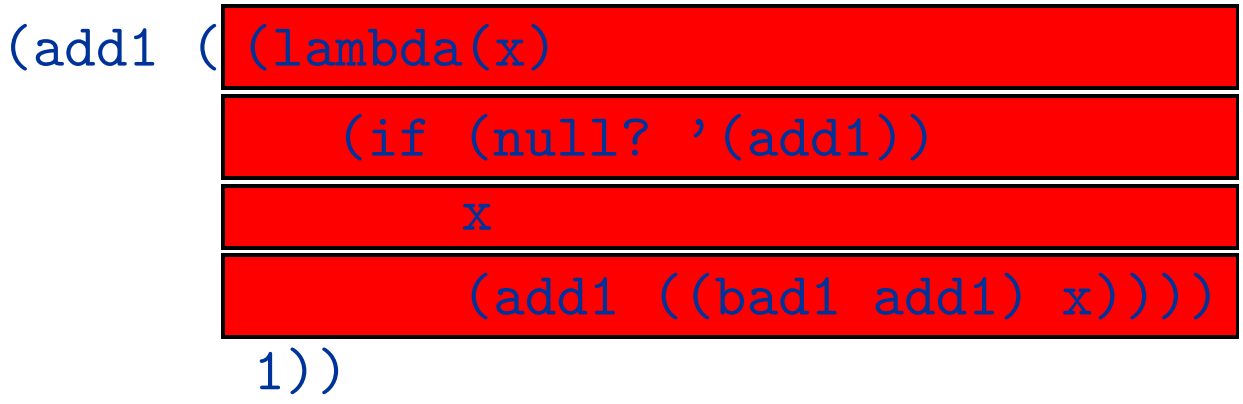
```
(add1 ((bad1 add1) 1))
```

$\xrightarrow{\beta}$

```
(add1 ((lambda(x)
         (if (null? '(add1))
             x
             (add1 ((bad1 add1) x))))
       1))
```

Étape suivante : λ ou if ou récursion ou forme normale

Application du résultat de bad1 (4)

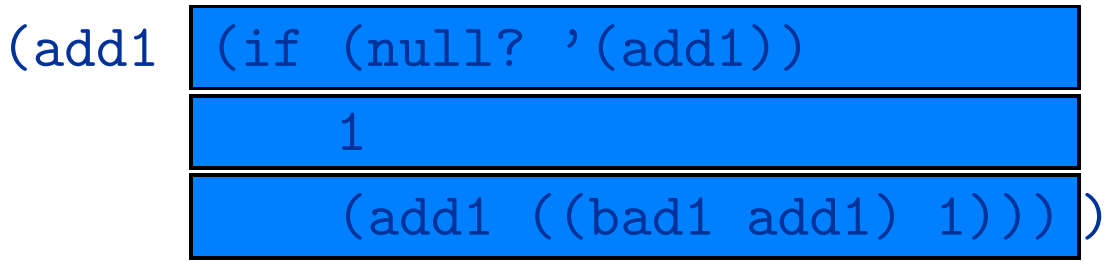


$\xrightarrow{\beta}$

```
(add1 (if (null? '(add1))
          1
          (add1 ((bad1 add1) 1))))
```

Étape suivante : λ ou if ou récursion ou forme normale

Application du résultat de bad1 (5)



$\xrightarrow{\beta}$

```
(add1 (add1 ((bad1 add1) 1)))
```

Étape suivante : λ ou if ou récursion ou forme normale

Application du résultat de bad1 (6)

```
(add1 (add1 ( (bad1 add1)  1)))
```

$\xrightarrow{\beta}$

```
(add1 (add1 ((lambda(x)
               (if (null? '())
                   x
                   (add1 ((bad1 add1) x))))
             1)))
```

Étape suivante : λ ou if ou récursion ou forme normale

Application du résultat de bad1 (7)

```
(add1 (add1 ( (lambda(x)
                (if (null? '())
                    x
                    (add1 ((bad1 add1) x))))
1)))
```

$\xrightarrow{\beta}$

```
(add1 (add1 (if (null? '())
                1
                (add1 ((bad1 add1))))))
```

Étape suivante : λ ou if ou récursion ou forme normale

Application du résultat de bad1 (8)

```
(add1 (add1 (if (null? '())
                1
                (add1 ((bad1 add1))))))
```

$\xrightarrow{\beta}$

```
(add1 (add1 1))
(add1 2)
3
```

Forme normale

Deuxième ébauche d'écriture : bad2

- On sort le if de la λ-expression afin qu'il s'éxécute lors de l'application de la fonctionnelle.

```
(define (bad2 . l)
  (if (null? l)
      identity
      (lambda (x)
        ((car l) ((apply bad2 (cdr l)) x)))))
```

- Application de bad2 :

```
> (define add2 (bad2 add1 add1))
```

```
(if (null? (list add1 add1))
    identity
    (lambda(x)
      (add1 ((bad2 add1) x))))
```

- **Étape suivante** : λ ou if ou récursion ou forme normale

Application de bad2 (1)

```
(if (null? (list add1 add1))
    identity
    (lambda(x)
      (add1 ((bad2 add1) x))))
```

$\xrightarrow{\beta}$

```
(lambda(x)
   (add1 ((bad2 add1) x)))
```

Étape suivante : λ ou if ou récursion ou forme normale

Application de bad2 (1)

```
(lambda(x)
   (add1 ((bad2 add1) x)))
```

Les calculs restants se déroulont lors de l'application de la fonction résultat `add2` : `(add2 1)`

```
((lambda(x)
   (add1 ((bad2 add1) x)))
 1)
```

Étape suivante : λ ou if ou récursion ou forme normale

Application de add2 (1)

```
((lambda(x)
    (add1 ((bad2 add1) x)))
 1)
```

$\xrightarrow{\beta}$

```
(add1 ((bad2 add1) 1))
```

Étape suivante : λ ou if ou récursion ou forme normale

Application de add2 (2)

```
(add1 ((bad2 add1) 1))
```

$\xrightarrow{\beta}$

```
(add1 ((if (null? (list add1))
           identity
           (lambda(x)
               (add1 ((bad2) x))))
       1))
```

Étape suivante : λ ou if ou récursion ou forme normale

Application de add2 (3)

```
(add1 ((if (null? (list add1))
           identity
           (lambda(x)
                 (add1 ((bad2) x))))
       1))
```

$\xrightarrow{\beta}$

```
(add1 ((lambda(x)
         (add1 ((bad2) x)))
       1))
```

Étape suivante : λ ou if ou récursion ou forme normale

Application de add2 (4)

```
(add1 ( (lambda(x)
          (add1 ((bad2) x)))
        1))
```

$\xrightarrow{\beta}$

```
(add1 (add1 ((bad2) 1)))
```

Étape suivante : λ ou if ou récursion ou forme normale

Application de add2 (5)

```
(add1 (add1 ( (bad2)  1)))
```

$\xrightarrow{\beta}$

```
(add1 (add1 ((if (null? '())
                 identity
                 (lambda(x) (add1 ((bad2) x))))
             1)))
```

Étape suivante : λ ou if ou récursion ou forme normale

Application de add2 (6)

```
(add1 (add1 ((if (null? '())
                 identity
                 (lambda(x) (add1 ((bad2) x))))
             1)))
```

$\xrightarrow{\beta}$

```
(add1 (add1 (identity 1)))
(add1 (add1 1))
(add1 2)
3
```

Forme normale

Application de **o**

```
> (define sqr1+ (o add1 sqr))
```

```
(if (null? (list sqr))
    add1
    (o (lambda(x)
         (add1 (sqr x)))))
```

Étape suivante : λ ou if ou récursion ou forme normale

Application de **o** (1)

```
(if (null? (list sqr))
    add1
    (o (lambda(x)
         (add1 (sqr x)))))
```

$\xrightarrow{\beta}$

```
(o (lambda(x)
     (add1 (sqr x))))
```

Étape suivante : λ ou if ou récursion ou forme normale

Application de **o** (2)

```
(o (lambda(x)
     (add1 (sqr x))))
```

$\xrightarrow{\beta}$

```
(if (null? '())
    (lambda(x)
      (add1 (sqr x)))
    (o (lambda(x) ...)))
```

Étape suivante : λ ou if ou récursion ou forme normale

Application de **o** (3)

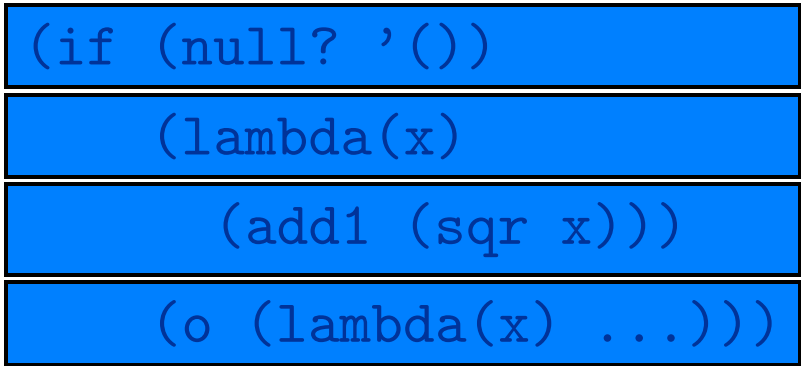
```
(if (null? '())
    (lambda(x)
      (add1 (sqr x)))
    (o (lambda(x) ...)))
```

$\xrightarrow{\beta}$

```
(lambda(x)
   (add1 (sqr x)))
```

Étape suivante : λ ou if ou récursion ou forme normale

Application de **o** (3)



$\xrightarrow{\beta}$

```
(lambda(x)
   (add1 (sqr x)))
```

forme normale

Application de **sqr1+**

Les calculs suivants sont réalisés lors de l'application de la fonction `sqr1+`. Ils ne concernent que ceux qui dépendent de son argument.

```
> (sqr1+ 1)
```

```
((lambda(x)
    (add1 (sqr x)))
  1)
```

$\xrightarrow{\beta}$

```
(add1 (sqr 1))
```