Recursion. Example $n! = 1 \cdot 2 \cdot \ldots \cdot n$, $n \geq 0$

Mathematical definition: recursion formula

$$
\begin{array}{rcll}
0! & = & 1 & (i) \\
n! & = & n \cdot (n-1)!, \quad \text{for } n > 0 & (ii)
\end{array}
$$

Computation:

$$
\begin{array}{cll}
 & 3! & \\
= & 3 \cdot (3-1)! & (ii) \\
= & 3 \cdot 2 \cdot (2-1)! & (ii) \\
= & 3 \cdot 2 \cdot 1 \cdot (1-1)! & (ii) \\
= & 3 \cdot 2 \cdot 1 \cdot 1 & (i) \\
= & 6 &
\end{array}
$$

Recursion. Example $n! = 1 \cdot 2 \cdot \ldots \cdot n$, $n \geq 0$

Mathematical definition: recursion formula

$$
\begin{array}{rcll}
0! & = & 1 & (i) \\
n! & = & n \cdot (n-1)!, \quad \text{for } n > 0 & (ii)
\end{array}
$$

Computation:

$$
\begin{array}{rll}
 & 3! & \\
= & 3 \cdot (3-1)! & (ii) \\
= & 3 \cdot 2 \cdot (2-1)! & (ii) \\
= & 3 \cdot 2 \cdot 1 \cdot (1-1)! & (ii) \\
= & 3 \cdot 2 \cdot 1 \cdot 1 & (i) \\
= & 6 &
\end{array}
$$

Recursive declaration. Example n!

Function declaration:

```
let rec fact = function
  | 0 -> 1                    (* i *)
  | n -> n * fact(n-1);;     (* ii *)
val fact : int -> int
```

Evaluation:

$$
\begin{array}{lll}
 & \mathtt{fact}(3) & \\
\rightsquigarrow & 3 * \mathtt{fact}(3-1) & (ii) \quad [n \mapsto 3] \\
\rightsquigarrow & 3 * 2 * \mathtt{fact}(2-1) & (ii) \quad [n \mapsto 2] \\
\rightsquigarrow & 3 * 2 * 1 * \mathtt{fact}(1-1) & (ii) \quad [n \mapsto 1] \\
\rightsquigarrow & 3 * 2 * 1 * 1 & (i) \quad [n \mapsto 0] \\
\rightsquigarrow & 6 &
\end{array}
$$

$e_1 \rightsquigarrow e_2$ reads: e_1 evaluates to e_2

Recursive declaration. Example n!

Function declaration:

```
let rec fact = function
  | 0 -> 1                      (* i *)
  | n -> n * fact(n-1);;        (* ii *)
val fact : int -> int
```

Evaluation:

$$
\begin{array}{lllll}
 & \texttt{fact}(3) & & \\
\leadsto & 3 * \texttt{fact}(3-1) & (ii) & [\texttt{n} \mapsto 3] \\
\leadsto & 3 * 2 * \texttt{fact}(2-1) & (ii) & [\texttt{n} \mapsto 2] \\
\leadsto & 3 * 2 * 1 * \texttt{fact}(1-1) & (ii) & [\texttt{n} \mapsto 1] \\
\leadsto & 3 * 2 * 1 * 1 & (i) & [\texttt{n} \mapsto 0] \\
\leadsto & 6 & &
\end{array}
$$

$e_1 \leadsto e_2$ reads: e_1 evaluates to e_2

Recursive declaration. Example $n!$

Function declaration:

```
let rec fact = function
  | 0 -> 1                  (* i *)
  | n -> n * fact(n-1);;    (* ii *)
val fact : int -> int
```

Evaluation:

$$
\begin{array}{lllr}
 & \mathtt{fact}(3) & & \\
\rightsquigarrow & 3 * \mathtt{fact}(3-1) & (ii) & [\mathtt{n} \mapsto 3] \\
\rightsquigarrow & 3 * 2 * \mathtt{fact}(2-1) & (ii) & [\mathtt{n} \mapsto 2] \\
\rightsquigarrow & 3 * 2 * 1 * \mathtt{fact}(1-1) & (ii) & [\mathtt{n} \mapsto 1] \\
\rightsquigarrow & 3 * 2 * 1 * 1 & (i) & [\mathtt{n} \mapsto 0] \\
\rightsquigarrow & 6 & &
\end{array}
$$

$e_1 \rightsquigarrow e_2$ reads: e_1 evaluates to e_2

Recursive declaration. Example $n!$

Function declaration:

```
let rec fact = function
  | 0 -> 1                    (* i *)
  | n -> n * fact(n-1);;     (* ii *)
val fact : int -> int
```

Evaluation:

$$
\begin{array}{lllll}
 & \mathtt{fact}(3) & & \\
\rightsquigarrow & 3 * \mathtt{fact}(3-1) & (ii) & [\mathtt{n} \mapsto 3] \\
\rightsquigarrow & 3 * 2 * \mathtt{fact}(2-1) & (ii) & [\mathtt{n} \mapsto 2] \\
\rightsquigarrow & 3 * 2 * 1 * \mathtt{fact}(1-1) & (ii) & [\mathtt{n} \mapsto 1] \\
\rightsquigarrow & 3 * 2 * 1 * 1 & (i) & [\mathtt{n} \mapsto 0] \\
\rightsquigarrow & 6 & &
\end{array}
$$

$e_1 \rightsquigarrow e_2$ reads: e_1 evaluates to e_2

Recursive declaration. Example n!

Function declaration:

```
let rec fact = function
  | 0 -> 1                    (* i *)
  | n -> n * fact(n-1);;      (* ii *)
val fact : int -> int
```

Evaluation:

$$
\begin{array}{lllr}
 & \mathtt{fact}(3) & & \\
\rightsquigarrow & 3 * \mathtt{fact}(3-1) & (ii) & [\mathtt{n} \mapsto 3] \\
\rightsquigarrow & 3 * 2 * \mathtt{fact}(2-1) & (ii) & [\mathtt{n} \mapsto 2] \\
\rightsquigarrow & 3 * 2 * 1 * \mathtt{fact}(1-1) & (ii) & [\mathtt{n} \mapsto 1] \\
\rightsquigarrow & 3 * 2 * 1 * 1 & (i) & [\mathtt{n} \mapsto 0] \\
\rightsquigarrow & 6 & &
\end{array}
$$

$e_1 \rightsquigarrow e_2$ reads: e_1 evaluates to e_2

Recursive declaration. Example n!

Function declaration:

```
let rec fact = function
  | 0 -> 1                  (* i *)
  | n -> n * fact(n-1);;   (* ii *)
val fact : int -> int
```

Evaluation:

$$
\begin{array}{llll}
 & \texttt{fact}(3) & & \\
\rightsquigarrow & 3 * \texttt{fact}(3-1) & (ii) & [\texttt{n} \mapsto 3] \\
\rightsquigarrow & 3 * 2 * \texttt{fact}(2-1) & (ii) & [\texttt{n} \mapsto 2] \\
\rightsquigarrow & 3 * 2 * 1 * \texttt{fact}(1-1) & (ii) & [\texttt{n} \mapsto 1] \\
\rightsquigarrow & 3 * 2 * 1 * 1 & (i) & [\texttt{n} \mapsto 0] \\
\rightsquigarrow & 6 & &
\end{array}
$$

$e_1 \rightsquigarrow e_2$ reads: e_1 evaluates to e_2

Recursion. Example $x^n = x \cdot \ldots \cdot x$, n occurrences of x

Mathematical definition: recursion formula

$$
\begin{array}{lcll}
x^0 & = & 1 & (1) \\
x^n & = & x \cdot x^{n-1}, \quad \text{for } n > 0 & (2)
\end{array}
$$

Function declaration:

```
let rec power = function
  | (_,0) -> 1.0                    (* 1 *)
  | (x,n) -> x * power(x,n-1)       (* 2 *)
```

Patterns:

`(_,0)` matches any pair of the form $(x, 0)$.
The wildcard pattern `_` matches any value.

`(x,n)` matches any pair (u, i) yielding the bindings

$$\mathtt{x} \mapsto u, \mathtt{n} \mapsto i$$

Recursion. Example $x^n = x \cdot \ldots \cdot x$, n occurrences of x

Mathematical definition: recursion formula

$$x^0 = 1 \quad (1)$$
$$x^n = x \cdot x^{n-1}, \quad \text{for } n > 0 \quad (2)$$

Function declaration:

```
let rec power = function
  | (_,0) -> 1.0                 (* 1 *)
  | (x,n) -> x * power(x,n-1)   (* 2 *)
```

Patterns:

$(\_, 0)$ matches any pair of the form $(x, 0)$.
The wildcard pattern _ matches any value.

(x, n) matches any pair (u, i) yielding the bindings

$$x \mapsto u, n \mapsto i$$

Recursion. Example $x^n = x \cdot \ldots \cdot x$, n occurrences of x

Mathematical definition: recursion formula

$$x^0 = 1 \qquad (1)$$
$$x^n = x \cdot x^{n-1}, \quad \text{for } n > 0 \qquad (2)$$

Function declaration:

```
let rec power = function
  | (_,0) -> 1.0                  (* 1 *)
  | (x,n) -> x * power(x,n-1)    (* 2 *)
```

Patterns:

`(_, 0)` matches any pair of the form $(x, 0)$.
The wildcard pattern `_` matches any value.

`(x, n)` matches any pair (u, i) yielding the bindings

$$\mathtt{x} \mapsto u, \mathtt{n} \mapsto i$$

Evaluation. Example: `power(4.0, 2)`

Function declaration:

```
let rec power = function
  | (_,0) -> 1.0                    (* 1 *)
  | (x,n) -> x * power(x,n-1)   (* 2 *)
```

Evaluation:

	$\mathtt{power}(4.0, 2)$	
$\rightsquigarrow$	$4.0 * \mathtt{power}(4.0, 2 - 1)$	Clause 2, $[\mathtt{x} \mapsto 4.0, \mathtt{n} \mapsto 2]$
$\rightsquigarrow$	$4.0 * \mathtt{power}(4.0, 1)$	
$\rightsquigarrow$	$4.0 * (4.0 * \mathtt{power}(4.0, 1 - 1))$	Clause 2, $[\mathtt{x} \mapsto 4.0, \mathtt{n} \mapsto 1]$
$\rightsquigarrow$	$4.0 * (4.0 * \mathtt{power}(4.0, 0))$	
$\rightsquigarrow$	$4.0 * (4.0 * 1)$	Clause 1
$\rightsquigarrow$	16.0	