

structural recursion

When defining a program to process an inductively-defined data type, the structure of the **program** should follow the structure of the **data**.

non-negative integers

JSON data

```
foo ::= bar
      | baz
      | (bif . foo)
```

```
foo ::= bar
      | baz
      | (bif . foo)
```

```
(cond ((bar? arg) ...)
      ((baz? arg) ...)
      (else ; cons bif foo
           ...))
```

a common patch:

interface procedure

1. Rename the function as a helper.
2. Write a new function that calls the helper.

<s-list>

::= ()

| (<symbol-expression> . <s-list>)

<symbol-expression>

::= <symbol>

| <s-list>

()

(a)

(a b c)

(a b c d)

(a b c d e f g h)

(())

((a) b)

(a (b) c)

(if (zero? n) zero (/ total n))

(cons (foo (first x)) (solve (rest x)))

```
(subst 'd 'b  
  '(a b c a b c d))
```

```
(subst 'a 'b  
  '((a b) ((b g r) (f r)) c (d e) b))
```

WARNING:

THIS IS A CAUTIONARY TALE

Patient: Doctor, it hurts when I do this.

Doctor: So, don't do that.

<s-list>

::= ()

| (<symbol-expression> . <s-list>)

<symbol-expression>

::= <symbol>

| <s-list>

mutual recursion