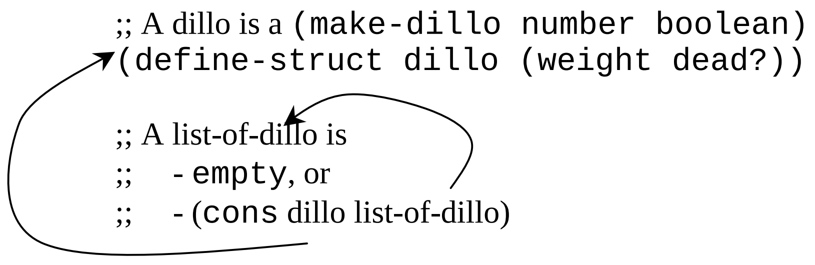
Data Definition Guidelines : Lists of Structures (or any non-atomic data)

Example: a list of dillos

Data definition format:



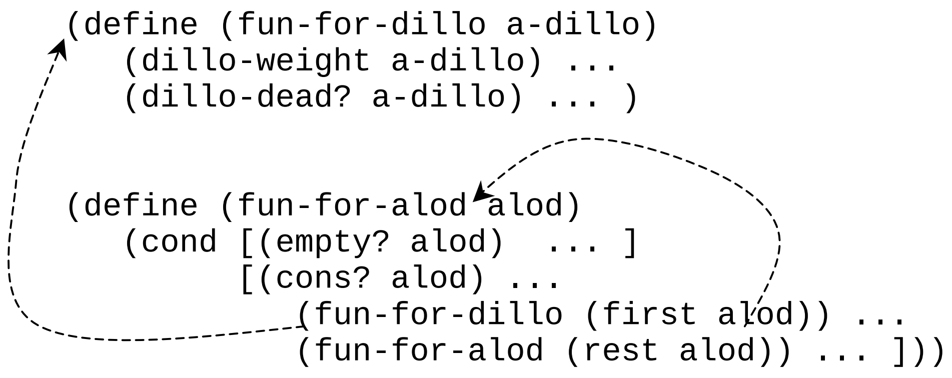
```
;; A dillo is a (make-dillo number boolean)
(define-struct dillo (weight dead?))

;; A list-of-dillo is
;;   - empty, or
;;   - (cons dillo list-of-dillo)
```

This definition has a similar format to that for lists of atomic data. All we've added is an arrow from the element of the list (dillo) to the data definition for the dillo. We do this because the list contains items of a type of data that we defined (as opposed to one that is built-in). As before, the arrows capture references between data definitions.

Template format: the template contains **one function template per data definition**. Each

- decides which case of data we have
- pulls out pieces in the cons case (the case that has pieces)
- uses calls to template functions to mimic each arrow in text



```
(define (fun-for-dillo a-dillo)
   (dillo-weight a-dillo) ...
   (dillo-dead? a-dillo) ... )


(define (fun-for-alod alod)
   (cond [(empty? alod)  ... ]
         [(cons? alod) ...
            (fun-for-dillo (first alod)) ...
            (fun-for-alod (rest alod)) ... ]))
```

Note we've taken a template for structures (`fun-for-dillo`) and a template for lists (`fun-for-alod`) and simply connected them with an arrow that mimics the one in the data definition. Except for the new arrow, we've made no changes to the templates for those two data definitions.

Remember: **the number of arrows in the data definition and template must always match**!