


Data Definition Guidelines : Trees

Example: a family tree with name, birth year, eye color, and parents for each person.

Data definition format:

```
;; A ft is
;; - 'unknown, or
;; - (make-person symbol number symbol ft ft)
(define-struct person (name year eye-color mother father))
```



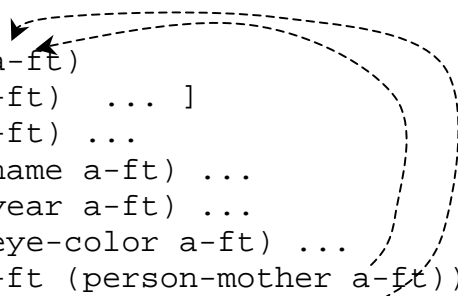
This definition has a similar format to that for lists, but there are two differences:

- We have two recursive arrows instead of one (since father and mother are both family trees)
- We don't use cons, since we're not building lists.
- We use 'unknown rather than empty, since we're not building lists

Template format: as before, the template

- decides which case of data we have
- pulls out pieces in appropriate cases
- uses recursive calls to mimic each arrow

```
(define (fun-for-ft a-ft)
  (cond [(symbol? a-ft) ... ]
        [(person? a-ft) ...
         (person-name a-ft) ...
         (person-year a-ft) ...
         (person-eye-color a-ft) ...
         (fun-for-ft (person-mother a-ft)) ...
         (fun-for-ft (person-father a-ft)) ... ]))
```



Note that the template doesn't use cons, since there is no cons in the data definition. Otherwise, this uses all of the same principles we've used to develop previous templates.