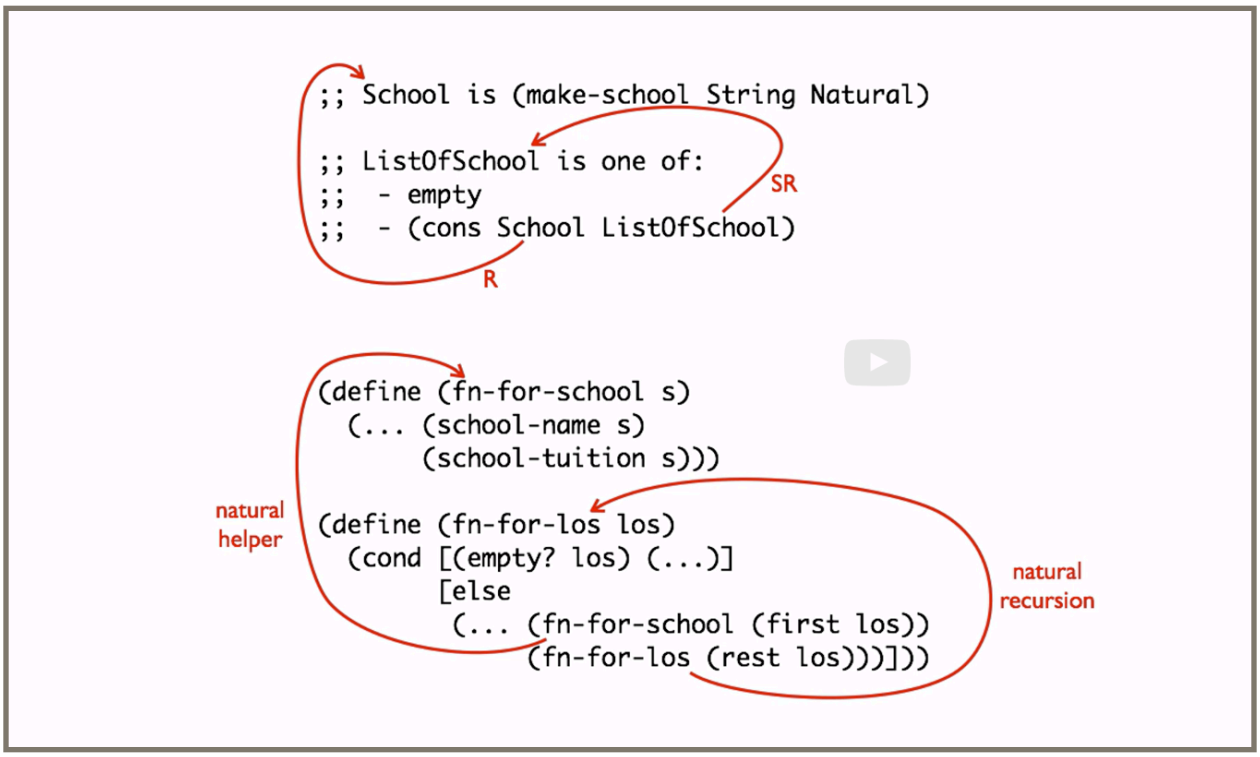
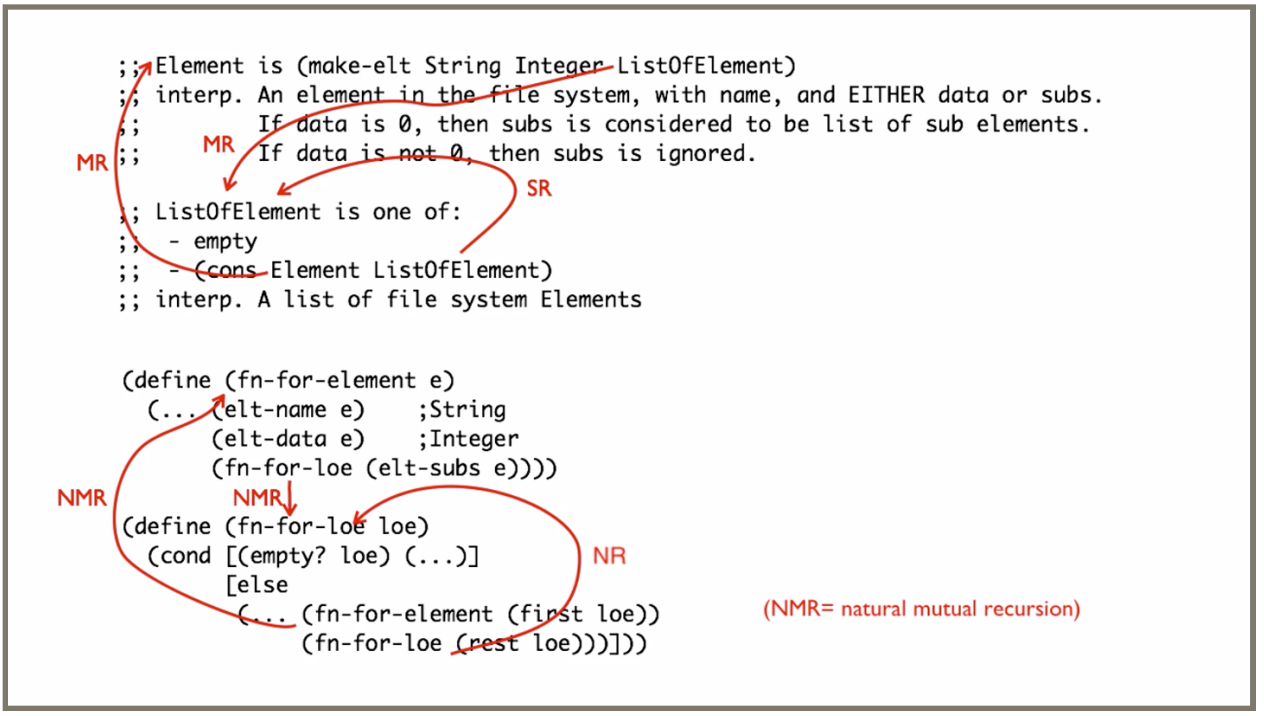
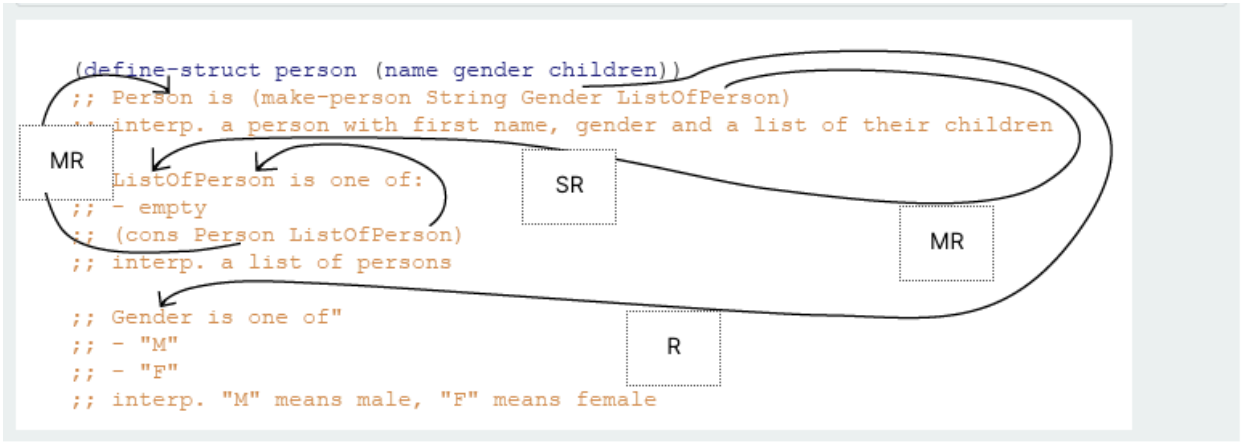
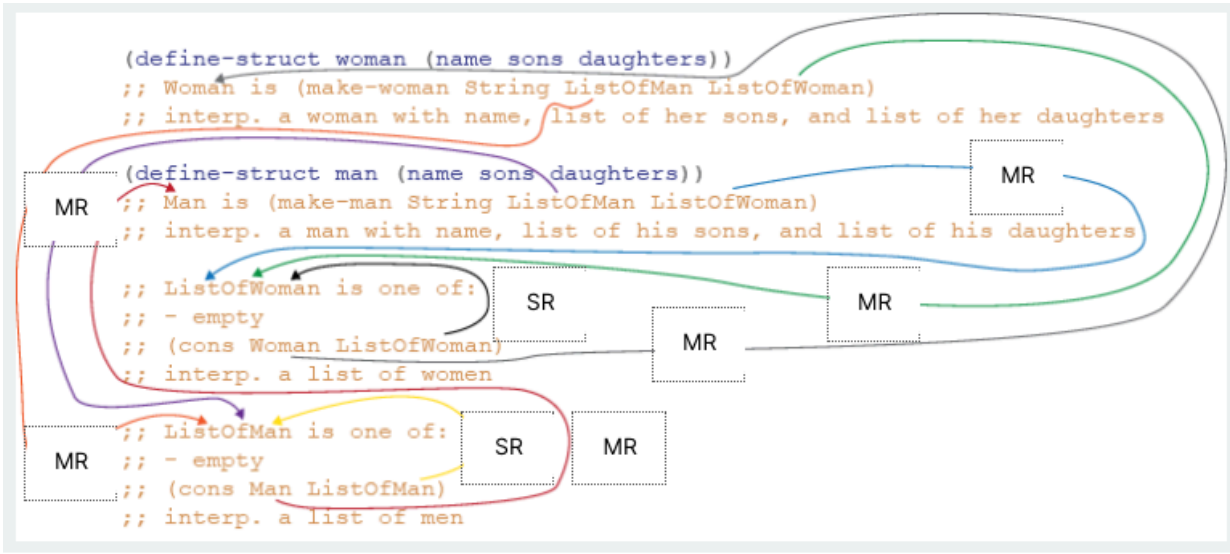

;; School is (make-school String Natural)
;; ListOfSchool is one of:
;; - empty
;; - (cons School ListOfSchool)
SR
R
(define (fn-for-school s)
  (... (school-name s)
       (school-tuition s)))
(define (fn-for-los los)
  (cond [(empty? los) (...)]
        [else
         (... (fn-for-school (first los))
              (fn-for-los (rest los)))]))
natural helper
natural recursion


;; Element is (make-elt String Integer ListOfElement)
;; interp. An element in the file system, with name, and EITHER data or subs.
;;   If data is 0, then subs is considered to be list of sub elements.
;;   If data is not 0, then subs is ignored.
;; ListOfElement is one of:
;; - empty
;; - (cons Element ListOfElement)
;; interp. A list of file system Elements
MR
MR
SR
(define (fn-for-element e)
  (... (elt-name e)   ;String
       (elt-data e)   ;Integer
       (fn-for-loe (elt-subs e))))
(define (fn-for-loe loe)
  (cond [(empty? loe) (...)]
        [else
         (... (fn-for-element (first loe))
              (fn-for-loe (rest loe)))]))
NMR
NMR
NR
(NMR= natural mutual recursion)


(define-struct person (name gender children))
;; Person is (make-person String Gender ListOfPerson)
;; interp. a person with first name, gender and a list of their children
;; ListOfPerson is one of:
;; - empty
;; (cons Person ListOfPerson)
;; interp. a list of persons
;; Gender is one of"
;; - "M"
;; - "F"
;; interp. "M" means male, "F" means female
MR
SR
MR
R


(define-struct woman (name sons daughters))
;; Woman is (make-woman String ListOfMan ListOfWoman)
;; interp. a woman with name, list of her sons, and list of her daughters
(define-struct man (name sons daughters))
;; Man is (make-man String ListOfMan ListOfWoman)
;; interp. a man with name, list of his sons, and list of his daughters
;; ListOfWoman is one of:
;; - empty
;; (cons Woman ListOfWoman)
;; interp. a list of women
;; ListOfMan is one of:
;; - empty
;; (cons Man ListOfMan)
;; interp. a list of men
MR
MR
SR
MR
MR
MR
SR
MR