

### Exercise 4.13.


Scheme allows us to create new bindings for variables by means of `define`, but provides no way to get rid of bindings. Implement for the evaluator a special form `make-unbound!` that removes the binding of a given symbol from the environment in which the `make-unbound!` expression is evaluated. This problem is not completely specified. For example, should we remove only the binding in the first frame of the environment? Complete the specification and justify any choices you make.

### Answer.

I think we should remove only the binding in the first frame of the environment, for it should keep consistent with the `define-variable!` operation. To remove a binding, we search the first frame for a variable of the binding, and remove the binding if it exist. If no such variable exists, we signal an “unbound variable” error.

```
(define (make-unbound! var env)
  (let ((frame (first-frame env)))
    (define (scan vars vals)
      (cond ((null? vars)
             (error "Unbound variable -- MAKE-UNBOUND!" var))
            ((eq? var (cadr vars))
             (begin (set-cdr! vars (cddr vars))
                    (set-cdr! vals (cddr vals))))
            (else (scan (cdr vars) (cdr vals)))))
      (scan (frame-variables frame)
            (frame-values frame))))
```

---

\*. Creative Commons  2014, Lawrence X. Amlord (颜世敏, aka 颜序).  
Email address: informlarry@gmail.com