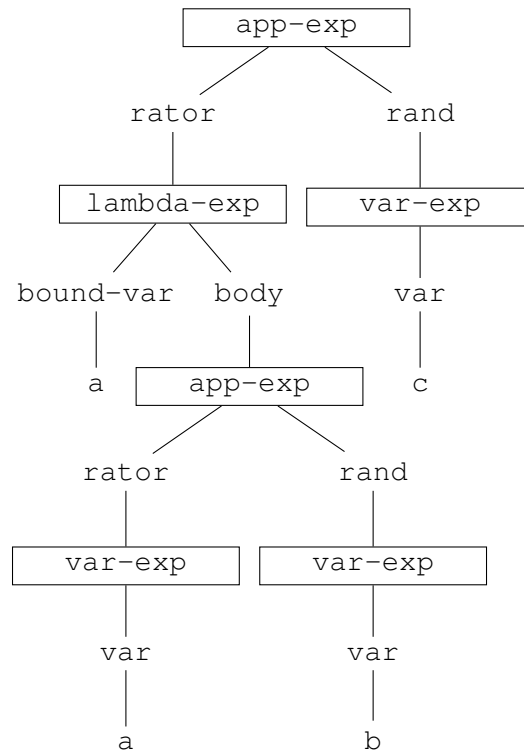


Abstract Syntax and Its Representation

Exercise 2.27



Exercise 2.28

```
(define unparse-lc-exp
  (lambda (exp)
    (cases lc-exp exp
      (var-exp (var)
        (symbol->string var))
      (lambda-exp (bound-var body)
        (string-append
          (string-append "(lambda ("
                          (symbol->string bound-var))
            (string-append ") "
              (unparse-lc-exp body)))))))
```

```

                                (unparse-lc-exp body))))
  (app-exp (rator rand)
    (string-append
      (string-append "("
        (unparse-lc-exp rator))
      (string-append
        (string-append " "
          (unparse-lc-exp rand))
        " "))))))

```

Exercise 2.29

```

(define-datatype lc-exp lc-exp?
  (var-exp
    (var identifier?))
  (lambda-exp
    (bound-vars (list-of identifier?))
    (body lc-exp?))
  (app-exp
    (rator lc-exp?)
    (rands (list-of lc-exp?))))

(define parse-expression
  (lambda (datum)
    (cond ((symbol? datum) (var-exp datum))
          ((pair? datum)
           (if (eqv? (car datum) 'lambda)
               (lambda-exp (map parse-expression (cadr datum))
                           (parse-expression (caddr datum)))
               (app-exp (parse-expression (car datum))
                        (map parse-expression (cdr datum)))))
          (else (report-invalid-concrete-syntax datum)))))

(define report-invalid-concrete-syntax

```

```

(lambda (datum)
  (eopl:error 'parse-expression
    "Could not parse ~s"
    datum)))

```

Exercise 2.30

```

(define parse-expression
  (lambda (sexp)
    (parse-s-exp sexp)))

(define parse-s-exp
  (lambda (sexp)
    (cond ((symbol? sexp)
      (var-exp sexp))
      ((pair? sexp)
      (parse-s-list sexp))
      (else (report-invalid-concrete-syntax sexp)))))

(define parse-s-list
  (lambda (slist)
    (cond ((null? slist)
      (report-empty-slist))
      ((lambda-syntax? slist)
      (lambda-exp (caadr slist)
        (parse-s-exp (caddr slist))))
      ((app-syntax? slist)
      (app-exp (parse-s-exp (car slist))
        (parse-s-exp (cadr slist))))
      (else (report-invalid-concrete-syntax slist)))))

(define lambda-syntax?
  (lambda (slist)
    (and (eqv? (car slist) 'lambda)

```

```

        (has-bound-variable-list? slist)
        (has-only-one-bound-variable? slist)
        (has-body? slist)
        (nothing-after-body? slist))))

(define has-bound-variable-list?
  (lambda (slist)
    (and (not (null? (cdr slist)))
         (pair? (cadr slist)))))

(define has-only-one-bound-variable?
  (lambda (slist)
    (null? (cdadr slist))))

(define has-body?
  (lambda (slist)
    (not (null? (cddr slist)))))

(define nothing-after-body?
  (lambda (slist)
    (null? (cddddr slist))))

(define app-syntax?
  (lambda (slist)
    (and (has-rand? slist)
         (nothing-after-rand? slist))))

(define has-rand?
  (lambda (slist)
    (not (null? (cdr slist)))))

(define nothing-after-rand?

```

```
(lambda (slist)
  (null? (cddr slist)))

(define report-empty-slist
  (lambda ()
    (eopl:error 'parse-s-list
      "Empty S-list")))
```

Exercise 2.31