```
;;; -*- Mode: Lisp; Syntax: Common-Lisp; -*-
;;; Code from Paradigms of Artificial Intelligence Programming
;;; Copyright (c) 1991 Peter Norvig

(defstruct op "An operation"
  (action nil) (preconds nil) (add-list nil) (del-list nil))

(defparameter *school-ops*
  (list
    (make-op :action 'drive-son-to-school
         :preconds '(son-at-home car-works)
         :add-list '(son-at-school)
         :del-list '(son-at-home))
    (make-op :action 'shop-installs-battery
         :preconds '(car-needs-battery shop-knows-problem shop-has-money)
         :add-list '(car-works))
    (make-op :action 'tell-shop-problem
         :preconds '(in-communication-with-shop)
         :add-list '(shop-knows-problem))
    (make-op :action 'telephone-shop
         :preconds '(know-phone-number)
         :add-list '(in-communication-with-shop))
    (make-op :action 'look-up-number
         :preconds '(have-phone-book)
         :add-list '(know-phone-number))
    (make-op :action 'give-shop-money
         :preconds '(have-money)
         :add-list '(shop-has-money)
         :del-list '(have-money))))
```

```
(defvar *state* nil "The current state: a list of conditions.")

(defvar *ops* nil "A list of available operators.")

(defun GPS (*state* goals *ops*)
  "General Problem Solver: achieve all goals using *ops*."
  (if (every #'achieve goals) 'solved))

(defun achieve (goal)
  "A goal is achieved if it already holds,
  or if there is an appropriate op for it that is applicable."
  (or (member goal *state*)
      (some #'apply-op
            (find-all goal *ops* :test #'appropriate-p))))

(defun appropriate-p (goal op)
  "An op is appropriate to a goal if it is in its add list."
  (member goal (op-add-list op)))

(defun apply-op (op)
  "Print a message and update *state* if op is applicable."
  (when (every #'achieve (op-preconds op))
    (print (list 'executing (op-action op)))
    (setf *state* (set-difference *state* (op-del-list op)))
    (setf *state* (union *state* (op-add-list op)))
    t))
```

```
* (gps '(son-at-home car-needs-battery have-money have-phone-book)
       '(son-at-school)
       *school-ops*)
(EXECUTING LOOK-UP-NUMBER)
(EXECUTING TELEPHONE-SHOP)
(EXECUTING TELL-SHOP-PROBLEM)
(EXECUTING GIVE-SHOP-MONEY)
(EXECUTING SHOP-INSTALLS-BATTERY)
(EXECUTING DRIVE-SON-TO-SCHOOL)
SOLVED

* (gps '(son-at-home car-needs-battery)
       '(son-at-school)
       *school-ops*)
NIL

* (gps '(son-at-home car-works)
       '(son-at-school)
       *school-ops*)
(EXECUTING DRIVE-SON-TO-SCHOOL)
SOLVED
```