

```

(in-package :user)

;;; Start with: (ipe '(goal (did-session)))

;;; TOP LEVELS

(def-operator do-session
  :goal (did-session)
  :filter ()
  :precond ()
  :recipe ((goal (did-beginning-of-session))
           (goal (did-load-input))
           (assert (is-step check-order))
           (goal (did-es-steps))
           (goal (did-end-of-session)))
  :hiercx ())

(def-operator do-beginning-of-session
  :goal (did-beginning-of-session)
  :filter ()
  :precond ()
  :recipe ((goal (did-utter "Welcome to the expert system simulator")))
  :hiercx ())

(def-operator do-load-input
  :goal (did-load-input)
  :filter ()
  :precond ()
  :recipe ((assert (to-be-bagged item01 bread plastic-bag medium no))
           (assert (to-be-bagged item02 glop jar small no))
           (assert (to-be-bagged item03 granola cardboard-box large no))
           (assert (to-be-bagged item04 ice-cream cardboard-carton
                       medium yes))
           (assert (to-be-bagged item05 potato-chips plastic-bag medium no)))
  :hiercx ())

(def-operator do-es-steps
  :goal (did-es-steps)
  :filter ()
  :precond ()
  :recipe ((goal (did-es-step))
           (fact (to-be-bagged ?itema ?namea ?containera ?sizea ?frozena))
           (retry-at (w-on-stack did-es-steps)))
  :hiercx ((assert (w-on-stack did-es-steps))))

(def-operator do-end-of-session
  :goal (did-end-of-session)
  :filter ()
  :precond ()
  :recipe ((goal (did-utter "Goodbye")))
  :hiercx ())

```

```
;;; *****
```

```
;;; EXPERT SYSTEM OPERATORS
```

```
(def-operator es-rule-default
  :goal (did-es-step)
  :filter ()
  :precond ((to-be-bagged ?itema ?namea ?containera ?sizea ?frozena))
  :recipe ((goal (did-utter ("Bagging" ?itema ": a" ?sizea
                             ?containera "of" ?namea)))
            (retract (to-be-bagged ?itema ?namea ?containera ?sizea ?frozena)))
  :hiercx ())
```

```
(def-operator es-rule-2
  :goal (did-es-step)
  :filter ()
  :precond ((is-step check-order))
  :recipe ((goal (did-utter "Bagging large items"))
            (retract (no-error (is-step ?x)))
            (assert (is-step bag-large-items)))
  :hiercx ())
```

```
(def-operator es-rule-1
  :goal (did-es-step)
  :filter ()
  :precond ((is-step check-order)
            (to-be-bagged ?itema potato-chips ?containera ?sizea ?frozena)
            (not (to-be-bagged ?itemb pepsi ?containerb ?sizeb ?frozenb)))
  :recipe ((goal (did-utter ("Chips without Pepsi--adding Pepsi"))
            (assert (to-be-bagged item-free-pepsi pepsi bottle large no)))
  :hiercx ())
```