File created: 17-Dec-87 08:36:13 **{POGO:AISNORTH:XEROX}<LOOPSCORE>LYRIC>USERS>TRUCKIN>TRUCKINV.;3**

changes to: (VARS TRUCKINVCOMS)

previous date: 14-Jan-85 16:57:14 {POGO:AISNORTH:XEROX}<LOOPSCORE>LYRIC>USERS>TRUCKIN>TRUCKINV.;2

Read Table: INTERLISP

Package: INTERLISP

Format: XCCS

```
;;
;; Copyright (c) 1985, 1987 by Xerox Corporation.  All rights reserved.

(RPAQQ TRUCKINVCOMS
       ((DECLARE%: DONTCOPY (PROP MAKEFILE-ENVIRONMENT TRUCKINV))
                                                                  ; Copyright (c) 1983 by Xerox Corporation.
        ;; Some primitive vocabulary functions for automatic TRUCKIN players. TRUCKIN is a domain for creating mini-expert system for teaching
        ;; knowledge representation techniques in the Loops programming system. Truckin provides a simple simulation environment for novice
        ;; Loops users in which small bodies of knowledge can be created and tested interactively. Knowledge in TRUCKIN is in the form of rules for
        ;; controlling a game piece to maximize profit.
        (CONSTANTS * TRUCKINVCONSTANTS)
        (FNS * TRUCKINVFNS)
        (FNS * StreamFns)
        (METHODS Player.AnyRoadStop Player.DirectionOf Player.Distance Player.FurthestRoadStop
               Player.NearestRoadStop Player.NthRoadStop Player.Range Player.Range1 Player.RoadStops
               Player.Sellers Player.TimeAtStop Player.TurnsAtStop)))

(DECLARE%: DONTCOPY

(PUTPROPS TRUCKINV MAKEFILE-ENVIRONMENT (:PACKAGE "IL" :READTABLE "INTERLISP" :BASE 10))
)

;; Copyright (c) 1983 by Xerox Corporation.
;; Some primitive vocabulary functions for automatic TRUCKIN players. TRUCKIN is a domain for creating mini-expert system for teaching knowledge
;; representation techniques in the Loops programming system. Truckin provides a simple simulation environment for novice Loops users in which small
;; bodies of knowledge can be created and tested interactively. Knowledge in TRUCKIN is in the form of rules for controlling a game piece to maximize
;; profit.

(RPAQQ TRUCKINVCONSTANTS (forwardSpellings))

(DECLARE%: EVAL@COMPILE

(RPAQQ forwardSpellings (F f Fwd fwd FWD Forward forward FORWARD))

(CONSTANTS forwardSpellings)
)

(RPAQQ TRUCKINVFNS (AnyBanditsP AnyRoadStop Buyers DirectionOf Distance FurthestRoadStop ISA NearestRoadStop
                          Nth NthRoadStop PricePerUnit RoadStops RoomToParkP SUBCLASS Sellers))

(DEFINEQ

(AnyBanditsP
  [LAMBDA (toRoadStop fromRoadStop)                                   (* edited%: "16-JUN-83 16:32")

          (* Returns T if there are any bandits between toRoadStop and fromRoadStop, and NIL otherwise.
          That is, NIL means it is SAFE. IF fromRoadStop is not given, then the roadStop of the currentPlayer is used.)

    (PROG (temp doneFlg banditFlg roadStop)
          [COND
             ((NULL fromRoadStop)
              (SETQ fromRoadStop (@ currentPlayer truck%:location]
          (COND
             ((NULL toRoadStop)
              (ERROR "Bad call to AnyBanditsP.  toRoadStop is  NIL.")))
          (COND
             ((ILESSP (@ toRoadStop milePost)
                      (@ fromRoadStop milePost))
              (SETQ temp toRoadStop)
              (SETQ toRoadStop fromRoadStop)
              (SETQ fromRoadStop temp)))
          (SETQ roadStop fromRoadStop)
          (WHILE (NOT (OR doneFlg banditFlg)) DO (SETQ banditFlg (_ roadStop BanditP))
                                                 (SETQ doneFlg (EQ roadStop toRoadStop))
                                                 (SETQ roadStop (@ roadStop next)))
          (RETURN banditFlg])

(AnyRoadStop
  [LAMBDA (roadStopType numMoves direction roomToParkFlg)             (* mjs%: "18-FEB-83 21:05")
```

(* * Randomly picks one of the RoadStops of type roadStopType, where roadStopType is the name of a super of some RoadStop. If numMoves is provided, it returns only those within that distance.
If direction='Fwd then only those in the forward direction, if direction='Bkwd then only those in the backward direction, if NIL then either direction. If roomToParkFlg is T, returns only those RoadStops that have room to park.)

```
(_ currentPlayer AnyRoadStop roadStopType numMoves direction roomToParkFlg])
```

(Buyers

```
[LAMBDA (commodityClass numMoves includeCDFlg)                (* mjs%: "18-FEB-83 21:05")
```

(* Returns all of the Buyers wishing to purchase the Commodity.
If numMoves is provided, it returns only those within that distance.
A common case is to use maxMove for numMoves. If includeCDFlg is T includes CityDumps also.)

```
(_ currentPlayer Buyers commodityClass numMoves includeCDFlg])
```

(DirectionOf

```
[LAMBDA (toRoadStop fromRoadStop)                             (* mjs%: "18-FEB-83 21:05")
```

(* * Returns direction of travel for going from fromRoadStop to toRoadStop.
If fromRoadStop is not given, then the location of the currentPlayer is assumed.)

```
(_ currentPlayer DirectionOf toRoadStop fromRoadStop])
```

(Distance

```
[LAMBDA (toRoadStop fromRoadStop)                             (* mjs%: "18-FEB-83 21:03")
```

(* * Computes the distance between fromRoadStop and toRoadStop.
If fromRoadStop is NIL, assumes the location of the currentPlayer.)

```
(_ currentPlayer Distance toRoadStop fromRoadStop])
```

(FurthestRoadStop

```
[LAMBDA (roadStops fromRoadStop)                              (* mjs%: "18-FEB-83 21:05")
```

(* * Returns the RoadStop furthest from fromRoadStop (excluding fromRoadStop) in the list of roadStops.
If fromRoadStop is not given, assumes the position of the currentPlayer.)

```
(_ currentPlayer FurthestRoadStop roadStops fromRoadStop])
```

(ISA

```
[LAMBDA (object className)                                    (* mjs%: "17-FEB-83 15:23")
```

(* * Procedural version of InstOf!. Hack to deal with Rule Compiler aversion to bangs.)

```
[COND
   ((LITATOM object)
    (SETQ object (GetObjectRec object]
(COND
   ((type? instance object)
    (_ object InstOf! className))
   (T (WRITE "Bad call to ISA. " object " not an instance."])
```

(NearestRoadStop

```
[LAMBDA (roadStops fromRoadStop)                              (* mjs%: "18-FEB-83 21:06")
```

(* * Returns the RoadStop nearest to fromRoadStop in the list of roadStops.
If fromRoadStop is not given, assumes the position of the currentPlayer.
Excludes fromRoadStop.)

```
(_ currentPlayer NearestRoadStop roadStops fromRoadStop])
```

(Nth

```
[LAMBDA (lst index)                                           (* mjs%: "26-JAN-83 16:38")
                                                              (* Returns the Nth item in a list.)
(CAR (NTH lst index])
```

(NthRoadStop

```
[LAMBDA (numMiles direction fromRoadStop roomToParkFlg)       (* mjs%: "18-FEB-83 21:06")
```

(* * Returns the Nth roadStop in the given direction from fromRoadStop.
If fromRoadStop is NIL, the location of the currentPlayer is used.
If direction is NIL, Forward is assumed. If there are fewer than numMiles roadStops in the specified direction, that is if the request would go off the highway, NthRoadStop returns the farthest RoadStop in that direction.)

```
(_ currentPlayer NthRoadStop numMiles direction fromRoadStop roomToParkFlg])
```

(PricePerUnit

```
  [LAMBDA (rs)                                                    (* edited%: " 2-AUG-83 14:26")
```

(* Returns the buying price per unit of the commodity at a producer roadstop.
If rs is not a producer, complains and returns 1)

```
    (COND
       ((_ rs InstOf! ($ Producer))
        (TIMES (@ rs pr)
               (@@ (@@ rs Commodity)
                   AvgPrice)))
       (T (printout TTY rs " is not a producer" T)
          1])
```

(RoadStops

```
  [LAMBDA (roadStopType numMoves direction roomToParkFlg)         (* sm%: " 1-JUL-83 18:03")
```

(* * Returns all of the RoadStops of type roadStopType, where roadStopType is the name of a super of some RoadStop.
If numMoves is provided, it returns only those within that distance.
If direction='Fwd then only those in the forward direction, if direction='Bkwd then only those in the backward direction, if NIL then either direction. If roomToParkFlg is T, returns only those RoadStops that have room to park.)

```
    (_ self RoadStops roadStopType numMoves direction roomToParkFlg])
```

(RoomToParkP

```
  [LAMBDA (roadStop)                                              (* mjs%: "18-FEB-83 12:33")
    (_ roadStop RoomToPark?])
```

(SUBCLASS

```
  [LAMBDA (class super)                                           (* mjs%: "17-FEB-83 12:28")
                                                                  (* Returns T if class is a sub-class of super)
    (SETQ class (COND
                   ((ATOM class)
                    (GetObjectRec class))
                   (T class)))
    (COND
       ((type? class class)
        (_ class Subclass super))
       (T (WRITE "Bad Arg to SUBCLASS. " class " not a class."])
```

(Sellers

```
  [LAMBDA (commodityClass numMoves)                               (* mjs%: "18-FEB-83 21:07")
```

(* * Returns all of the Sellers having the Commodity for sale. If numMoves is provided, it returns only those within that distance. A common case is to use maxMove for numMoves.)

```
    (_ currentPlayer Sellers commodityClass numMoves])

)

(RPAQQ StreamFns (FilterObjs PickHiObj PickLowObj SortObjs))

(DEFINEQ
```

(FilterObjs

```
  [LAMBDA (self selector objects)                                 (* mjs%: " 9-JUN-83 23:22")
```

(* FilterObjs sends a selector msg to self for each object in the list of objects and returns all of the objects for which a non-NIL value is returned.)

```
    (for object in objects when (_!
                                 self selector object)
       collect object])
```

(PickHiObj

```
  [LAMBDA (self selector objects)                                 (* mjs%: " 9-JUN-83 23:23")
```

(* PickHiObj sends a selector msg to self for each object in the list of objects to determine a numeric rating for each of the objects. PickHiObj then returns the object with the highest rating.
When the value is non-numeric, the corresponding object is excluded.)

```
    (PROG (objectRatingPairs hiRating hiObject)
          (SETQ hiRating (FMAX))                                  (* Get ratings.)
          [SETQ objectRatingPairs (for object in objects collect (CONS object (_!
                                                                               self selector object]
                                                                  (* Prune out non-numeric ratings.)
          (SETQ objectRatingPairs (for pair in objectRatingPairs when (NUMBERP (CDR pair)) collect pair))
                                                                  (* Select the highest rated object.)
          (for pair in objectRatingPairs when (GREATERP (CDR pair)
                                                        hiRating)
             do (SETQ hiRating (CDR pair))
                (SETQ hiObject (CAR pair)))
          (RETURN hiObject])
```

(PickLowObj

```
  [LAMBDA (self selector objects)                                    (* mjs%: " 9-JUN-83 23:24")

          (* PickLowObj sends a selector message to self for each object in the list of objects to determine a numeric rating for each
          of the objects. PickLowObj then returns the object with the lowest rating.
          When the value is non-numeric, the corresponding object is excluded.)

    (PROG (objectRatingPairs lowRating lowObject)
          (SETQ lowRating (FMIN))                                    (* Get ratings.)
          [SETQ objectRatingPairs (for object in objects collect (CONS object (_!
                                                                        self selector object]
                                                                     (* Prune out non-numeric ratings.)
          (SETQ objectRatingPairs (for pair in objectRatingPairs when (NUMBERP (CDR pair)) collect pair))
                                                                     (* Select the lowest rated object.)
          (for pair in objectRatingPairs when (LESSP (CDR pair)
                                                     lowRating)
             do (SETQ lowRating (CDR pair))
                (SETQ lowObject (CAR pair)))
          (RETURN lowObject])
```

(SortObjs

```
  [LAMBDA (self selector objects)                                    (* mjs%: " 9-JUN-83 23:26")

          (* SortObjs sends a selector message to self for to each object in the list of objects to determine a numeric rating for each of
          the objects. SortObjs then returns a list of the objects in descending order.
          If the value for an object is non-numeric, the corresponding object is excluded.)

    (PROG (objectRatingPairs)                                        (* Get ratings.)
          [SETQ objectRatingPairs (for object in objects collect (CONS object (_!
                                                                        self selector object]
                                                                     (* Prune out non-numeric ratings.)
          (SETQ objectRatingPairs (for pair in objectRatingPairs when (NUMBERP (CDR pair)) collect pair))
                                                                     (* Sort the pairs.)
          [SORT objectRatingPairs (FUNCTION (LAMBDA (x y)
                                            (GREATERP (CDR x)
                                                      (CDR y]        (* Collect the objects.)
          (SETQ objects (for pair in objectRatingPairs collect (CAR pair)))
          (RETURN objects])

)

(\BatchMethodDefs)

(Method ((Player AnyRoadStop) self roadStopType numMoves direction roomToParkFlg)
                                                                     ; sm: 11-FEB-83 11:49
   ;; Randomly picks one of the RoadStops of type roadStopType, where roadStopType is the name of a super of some RoadStop.
   "If numMoves is provided, it returns only those within that distance. If direction='Fwd then only those in
   the forward direction, if direction='Bkwd then only those in the backward direction, if NIL then either
   direction. If roomToParkFlg is T, returns only those RoadStops that have room to park."
   [PROG (roadStops)
         (SETQ roadStops (RoadStops roadStopType numMoves direction roomToParkFlg))
         (RETURN (COND
                    ((NULL (CADR roadStops))                         (* special case if only 1)
                     (CAR roadStops))
                    (T                                               (* Here for random selection.)
                       (Nth roadStops (RAND 1 (FLENGTH roadStops])

(Method ((Player DirectionOf) self toRoadStop fromRoadStop)          ; sm: 11-FEB-83 11:53
   "Returns direction of travel for going from fromRoadStop to toRoadStop. If fromRoadStop is not given, then
   the location of the currentPlayer is assumed."
   [COND
      ((NULL fromRoadStop)
       (SETQ fromRoadStop (@ (@ truck)
                             location]
   (COND
      ((IGREATERP (@ toRoadStop milePost)
                  (@ fromRoadStop milePost))
       'Forward)
      (T 'BackWard)))

(Method ((Player Distance) self toRoadStop fromRoadStop)             ; sm: 11-FEB-83 11:54
   "Computes the distance between fromRoadStop and toRoadStop. If fromRoadStop is NIL, assumes the location of
   the currentPlayer."
   [COND
      ((NULL fromRoadStop)
       (SETQ fromRoadStop (@ (@ truck)
                             location]
   (IABS (IDIFFERENCE (@ fromRoadStop milePost)
                      (@ toRoadStop milePost))))

(Method ((Player FurthestRoadStop) self roadStops fromRoadStop)      ; sm: 11-FEB-83 11:56
   "Returns the RoadStop furthest from fromRoadStop (excluding fromRoadStop) in the list of roadStops. If
   fromRoadStop is not given, assumes the position of the currentPlayer."
```

```
   (PROG ((maxDistance -1)
          distance farRS)

         (* * Default to current position.)

         [COND
            ((NULL fromRoadStop)
             (SETQ fromRoadStop (@ (@ truck)
                                   location]

         (* * Find furthest excluding fromRoadStop.)

         (for rs in roadStops when (AND (NEQ rs fromRoadStop)
                                        (IGREATERP (SETQ distance (Distance fromRoadStop rs))
                                              maxDistance))
            do (SETQ maxDistance distance)
               (SETQ farRS rs))
         (RETURN farRS)))

(Method ((Player NearestRoadStop) self roadStops fromRoadStop) ; sm: 11-FEB-83 11:59
                                                               ; Returns the RoadStop nearest to fromRoadStop in the list of
                                                               ; roadStops.
   "If fromRoadStop is not given, assumes the position of the currentPlayer. Excludes fromRoadStop."
   (PROG ((minDistance 1000)
          distance nearRS)

         (* * Default to current position.)

         [COND
            ((NULL fromRoadStop)
             (SETQ fromRoadStop (@ (@ truck)
                                   location]

         (* * Find nearest excluding fromRoadStop.)

         (for rs in roadStops when (AND (NEQ rs fromRoadStop)
                                        (ILESSP (SETQ distance (Distance fromRoadStop rs))
                                              minDistance))
            do (SETQ minDistance distance)
               (SETQ nearRS rs))
         (RETURN nearRS)))

(Method ((Player NthRoadStop) self numMiles direction fromRoadStop roomToParkFlg)
                                                               ; sm:  5-JUL-83 17:38
   ;; Returns the Nth roadStop in the given direction from fromRoadStop. If roomToParkFlg, it picks the farthest RoadStop under numMiles in which
   ;; there is room. If no room at all, returns fromRoadStop.
   "If fromRoadStop is NIL, the location of the currentPlayer is used. If direction is NIL, Forward is assumed.
   If there are fewer than numMiles roadStops in the specified direction, that is if the request would go off
   the highway, NthRoadStop returns the farthest RoadStop in that direction."
   (PROG (rs)
         [COND
            ((NULL direction)
             (SETQ direction 'Forward]
         [COND
            ((NULL fromRoadStop)
             (SETQ fromRoadStop (@ (@ truck)
                                   location]
         (COND
            [(NULL numMiles)
             (SETQ numMiles (FLENGTH (@ Simulator roadStops]
            ((ZEROP numMiles)
             (RETURN fromRoadStop)))
     Loop
         (SETQ rs fromRoadStop)
         [for k from 1 to numMiles do (SETQ rs (COND
                                                  ((FMEMB direction forwardSpellings)
                                                   (OR (@ rs next)
                                                       rs))
                                                  (T (OR (@ rs prev)
                                                         rs]
         (COND
            ((AND roomToParkFlg (NULL (_ rs RoomToPark?)))       (* Try again if no room to park.)
             (SETQ numMiles (SUB1 numMiles))
             (COND
                ((ILEQ numMiles 0)
                 (RETURN fromRoadStop)))
             (GO Loop)))
         (RETURN rs)))

(Method ((Player Range) self)                                  ; mjs: 26-JAN-83 16:54
   "Computers the travel range for the current player as limited only by fuel."
   [PROG ((truck (@ truck)))
         (RETURN (IQUOTIENT (@ truck fuel)
                        (@@ truck Gpm])

(Method ((Player Range1) self)                                 ; sm:  5-JUL-83 17:17
```

```
   "Computers the travel range in a single move for the current player as limited only by fuel maxMove."
   [PROG ((truck (@ truck)))
         (RETURN (MIN (@ maxMove)
                      (IQUOTIENT (@ truck fuel)
                             (@@ truck Gpm])

(Method ((Player RoadStops) self roadStopType numMoves direction roomToParkFlg)
                                                                  ; sm: 5-JUL-83 17:39
   ;; Returns all of the RoadStops of type roadStopType, where roadStopType is the name of a super of some RoadStop.
   "If numMoves is provided, it returns only those within that distance. If direction='Fwd then only those in
   the forward direction, if direction='Bkwd then only those in the backward direction, if NIL then either
   direction. If roomToParkFlg is T, returns only those RoadStops that have room to park."
   [PROG (currentPos startPos stopPos rs roadStops commodity (allRoadStops (@ Simulator roadStops)))
         [COND
            ((NULL roadStopType)                                  (* default to all roadstops.)
             (SETQ roadStopType 'RoadStop]
         (COND
            ((NULL (GetClassRec roadStopType))
             (WRITE "Bad call to RoadStops.  roadStopType unrecognized: " roadStopType)))
         [COND
            ((NULL numMoves)
             (SETQ numMoves (FLENGTH allRoadStops]

          (* * Compute search range using direction.)

         (SETQ currentPos (@ (@ (@ truck)
                                location)
                             milePost))
         [SETQ startPos (COND
                           ((FMEMB direction forwardSpellings)
                            currentPos)
                           (T (MAX 1 (IDIFFERENCE currentPos numMoves]
         [SETQ stopPos (COND
                          ((AND direction (NOT (FMEMB direction forwardSpellings)))
                           currentPos)
                          (T (MIN (FLENGTH allRoadStops)
                                  (IPLUS currentPos numMoves]

          (* * Make a list of the relevant roadstops.)

         (SETQ rs (CAR (NTH allRoadStops startPos)))
         (for pos from startPos to stopPos do (COND
                                                 ((AND (_ (Class rs)
                                                          Subclass roadStopType)
                                                       (OR (NULL roomToParkFlg)
                                                           (_ rs RoomToPark?)))
                                                  (push roadStops rs)))
                                              (SETQ rs (@ rs next)))
         (RETURN (COND
                    ((NOT (FMEMB direction forwardSpellings))
                     (DREVERSE roadStops))
                    (T roadStops])

(Method ((Player Sellers) self commodityClass numMoves)           ; edited: 2-AUG-83 14:26
                                                                  ; Returns all of the Sellers having the Commodity for sale.
   "If numMoves is provided, it returns only those within that distance. A common case is to use maxMove for
   numMoves."
   (PROG (currentPos startPos stopPos rs roadStops commodity (allRoadStops (@ Simulator roadStops)))
         [COND
            ((NULL (GetObjectRec commodityClass))                 (* default to all Commodities.)
             (printout TTY "commodityClass in SELLERS is NIL. Assuming $Commodity" T)
             (SETQ commodityClass ($ Commodity)))
            (T (SETQ commodityClass (GetObjectRec commodityClass]
         [COND
            ((NULL numMoves)
             (SETQ numMoves (FLENGTH allRoadStops]

          (* * Compute search range.)

         (SETQ currentPos (@ (@ (@ truck)
                                location)
                             milePost))
         (SETQ startPos (MAX 1 (IDIFFERENCE currentPos numMoves)))
         (SETQ stopPos (MIN (FLENGTH allRoadStops)
                            (IPLUS currentPos numMoves)))
         (SETQ rs (CAR (NTH allRoadStops startPos)))

          (* * Make a list of the relevant Producers.)

         (for pos from startPos to stopPos do (COND
                                                 ([AND (_ rs InstOf! 'Producer)
                                                       (SETQ commodity (@@ rs Commodity))
                                                       (_ commodity Subclass commodityClass)
                                                       (NOT (ZEROP (@ rs qty]
                                                  (push roadStops rs)))
                                              (SETQ rs (@ rs next)))
```

```
         (RETURN roadStops)))

(Method ((Player TimeAtStop) self)                                    ; sm: 18-MAY-83 11:02
   "Returns the time spent by player at the stop where currently parked."
   [PROG ((truck (@ truck))
          loc)
         (SETQ loc (@ truck location))
         (COND
            ((_ loc InstOf! 'RoadStop)
             (RETURN (_ loc TimeSpent self])

(Method ((Player TurnsAtStop) self)                                   ; sm: 19-MAY-83 15:13
   "how many turns parked at current location"
   [PROG ((truck (@ truck))
          loc)
         (SETQ loc (@ truck location))
         (COND
            ((_ loc InstOf! 'RoadStop)
             (RETURN (_ loc TurnsStayed self])

(\UnbatchMethodDefs)

(PUTPROPS TRUCKINV COPYRIGHT ("Xerox Corporation" 1985 1987))
```

FUNCTION INDEX

VARIABLE INDEX

CONSTANT INDEX

PROPERTY INDEX