

(FILECREATED " 6-JUL-83 16:28:40" {INDIGO}<LOOPS>TRUCKIN>MULTI>TRUCKINV.;3 23704

changes to: (VARS TRUCKINVFNs)

previous date: " 5-JUL-83 17:47:06" {INDIGO}<LOOPS>TRUCKIN>MULTI>TRUCKINV.;2)

(PRETTYCOMPRINT TRUCKINVCOMs)

(RPAQQ TRUCKINVCOMs ((* 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 * QueryFns)

(FNS * StreamFns)))

(* 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 (1st index) (* mjs: "26-JAN-83 16:38")

(* Returns the Nth item in a list.)

(CAR (NTH 1st 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)

(* mjs: "15-FEB-83 17:44")

(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 T 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)
 (← roadStop RoomToPark?)

(* mjs: "18-FEB-83 12:33")

(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 QueryFns (Player.AnyRoadStop Player.DirectionOf Player.Distance Player.FurthestRoadStop
 Player.NearestRoadStop Player.NthRoadStop Player.Range
 Player.Range1 Player.RoadStops Player.Sellers Player.TimeAtStop
 Player.TurnsAtStop))

(DEFINEQ

(Player.AnyRoadStop

[LAMBDA (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)))))
```

(Player.DirectionOf

```
[LAMBDA (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))
     (QUOTE Forward))
    (T (QUOTE BackWard]))
```

(Player.Distance

```
[LAMBDA (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)))
```

(Player.FurthestRoadStop

```
[LAMBDA (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])
```

(Player.NearestRoadStop

```
[LAMBDA (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])
```

(Player.NthRoadStop

```
[LAMBDA (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 (QUOTE 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])
  (RETURN rs])
```

(Player.Range

```
[LAMBDA (self)
```

(* mjs: "26-JAN-83 16:54")

(* Computes the travel range for the current player as limited only by fuel.)

```
(PROG ((truck (%@ truck)))
      (RETURN (IQUOTIENT (%@ truck fuel)
                        (%@ truck Gpm])))
```

(Player.Range1

[LAMBDA (self)

(* sm: " 5-JUL-83 17:17")

(* Computes 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]))
```

(Player.RoadStops

[LAMBDA (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)
      (SETQ roadStopType (QUOTE RoadStop) (* default to all roadstops.)
    (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]
```

(Player.Sellers

[LAMBDA (self commodityClass numMoves)

(* sm: " 5-JUL-83 17:40")

(* 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 T "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!(QUOTE Producer))
      (SETQ commodity (@@ rs Commodity))
      (+ commodity Subclass commodityClass)
      (NOT (ZEROP (@ rs qty)
        (push roadStops rs)))
      (SETQ rs (@ rs next)))
    (RETURN roadStops])
```

(Player.TimeAtStop
[LAMBDA (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!(QUOTE RoadStop))
     (RETURN (+ loc TimeSpent self]))
```

(Player.TurnsAtStop
[LAMBDA (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!(QUOTE RoadStop))
     (RETURN (+ loc TurnsStayed self]))
  )
```

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

```
)
(DECLARE: DONTCOPY
 (FILEMAP (NIL (1798 8655 (AnyBanditsP 1808 . 2836) (AnyRoadStop 2838 . 3489) (Buyers 3491 . 3912) (
 DirectionOf 3914 . 4263) (Distance 4265 . 4583) (FurthestRoadStop 4585 . 4963) (ISA 4965 . 5381) (
 NearestRoadStop 5383 . 5755) (Nth 5757 . 5984) (NthRoadStop 5986 . 6578) (PricePerUnit 6580 . 7090) (
 RoadStops 7092 . 7722) (RoomToParkP 7724 . 7862) (SUBCLASS 7864 . 8288) (Sellers 8290 . 8653)) (8923
 19762 (Player.AnyRoadStop 8933 . 10041) (Player.DirectionOf 10043 . 10561) (Player.Distance 10563 .
 11008) (Player.FurthestRoadStop 11010 . 11843) (Player.NearestRoadStop 11845 . 12811) (
 Player.NthRoadStop 12813 . 14349) (Player.Range 14351 . 14723) (Player.Range1 14725 . 15142) (
 Player.RoadStops 15144 . 17272) (Player.Sellers 17274 . 18919) (Player.TimeAtStop 18921 . 19360) (
 Player.TurnsAtStop 19362 . 19760)) (19829 23682 (FilterObjs 19839 . 20228) (PickHiObj 20230 . 21364) (
 PickLowObj 21366 . 22513) (SortObj 22515 . 23680))))))
STOP
```