Ma Zhencai Jayden
U087139J
Z model of Carpark

Carpark

carcount : N
lots : ParkingLot
cars : Car
parkAt : Car ↔ ParkingLot
currentTime: N
entryTime: Car ↔ N
parkedForDays : Car ↔ N

lot = 4

Carpark $_{INIT}$

carcount = 0
lots = {lot1, lot2, lot3, lot4}
cars = {}
currentTime = 0

Entering

Δ Carpark
c? : Car

carcount < 4
carcount' = carcount + 1
entryTime' = entryTine U {c? ↔ currentTime}
parkedForDays' = parkedForDays U {c? ↔ 0}
cars' = car U {c?}

Parking

Δ Carpark
c? : Car
p? : ParkingLot

c? ∈ cars
{ _↔p?} ∉ parkAt
parkAt' = parkAt U {c? ↔ p?}

UnParking

Δ Carpark
c? : Car
p? : ParkingLot

c? ∈ cars
{ c?↔p?} ∈ parkAt
parkAt' = parkAt – {c? ↔ p?}

Exiting

Δ Carpark
c? : Car
payment? : N
change! : N

c? ∈ cars
{c? ↔_ } ∉ parkAt
if payParkingFee(payment?) == -1
then
    change! = payment?
else
    change! = payParkingFee(payment?)
    carcount ' = carcount – 1
    cars' = car – c?

Clocking

Δ Carpark

if currentTime <=24
then
    currentTime' = currentTime +1
else
    currentTime' = 0
    ∀ y: x days y ● y = y + 24

parkedForDays : Car ↔ N

∀ parkedForDays : Car ↔ N●
dom parkedForDays = { x:Car ; y:N | x parkedForDays y ● x } Λ
ran parkedForDays = { x:Car ; y:N | x parkedForDays y ● y }

entryTime: Car ↔ N

∀ entryTime : Car ↔ N●
dom entryTime = { x:Car ; y:N | x entryTime y ● x } Λ
ran entryTime = { x:Car ; y:N | x entryTime y ● y }

parkAt : Car ↔ ParkingLot

∀ parkAt : Car ↔ ParkingLot ●
dom parkAt = { x:Car ; y:ParkingLot | x parkAt y ● x } Λ
ran parkAt = { x:Car ; y:ParkingLot | x parkAt y ● y }

payParkingFee: Car →N
c? : Car
payment? : N
change! : N

if payment? >= parkingFee(c?)
then
    change! = payment? – parkingFee(c?)
else
    change! = -1

parkingFee: Car →N
c? : Car
fee! : N

∃y ∃z: (c? entryTime y)Λ(c? parkedForDays z)Λ(c? ∈ cars)
fee! = 2 * (y – currentTime + z)