```
fact numberOfState{
     #Car = #State
}

fact passengersConstraint{
     all c:Car| ((c.state.phase=used) implies
(c.state.passengers<=c.seats)) and((c.state.phase!=used)
implies (c.state.passengers=0))
}

//this means that there are not cars in the same position
fact differentPositionInCar{
     all c1,c2 : Car | (c1!=c2) implies
(c1.state.statePosition != c2.state.statePosition)
}

----------------Phase part----------------

//these are the possible internal state of the car
enum Phase {reserved, used, charge, free,
parkUnsafeOrChargeOnSite}

fact reservedPhase{
     all c: Car| (c.state.phase = reserved) iff (c.isInSafe
and c.state.batteryLevel>1 and c.isUsed and
(c.state.phase!=used and c.state.phase!=charge and
c.state.phase!=free and
c.state.phase!=parkUnsafeOrChargeOnSite))
}

fact usedPhase{
     all c: Car| (c.state.phase = used) iff (c.isUsed and
c.state.batteryLevel != 0  and (c.state.phase!=reserved
and c.state.phase!=charge and c.state.phase!=free and
c.state.phase!=parkUnsafeOrChargeOnSite))
}

fact chargePhase{
     all c: Car| (c.state.phase = charge) iff
(!(c.isAvailable) and c.isInSpecial and
(c.state.phase!=reserved and c.state.phase!=used and
```