判题器逻辑结构

判题器的总体框架如图1所示，对于每个时间片而言有三个操作，首先让每条道路上的车向前行驶，并标记行驶后的状态（等待或终止），然后对每一个路口需要转弯的车进行转弯操作，最后车库中的车上路。

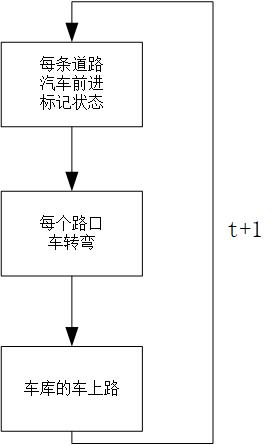


图1：判题器总框架

Object-Car:

Id; //车辆Id

From; //车辆起始点cross Id

To; //车辆终点cross Id

Speed; //车辆速度

State; //车辆状态，共四个状态，在车库/终止/等待/已到达目的地

Route； //车辆规划的路径

curRoadId; //当前道路Id

curRoadIndex; //当前道路在规划路径的第几个

roadDir; //当前道路方向，道路一般有两个方向

channel; //当前道路通道

dist; //当前道路通道距离最前端的距离

dir; //出路口方向，直行，左转或右转

planTime; //计划出发时间

realTime; //实际出发时间

Object-Road

属性：

Id; //道路Id

crossFrom; //道路起始cross

crossTo; //道路终止cross

Speed; //道路限速

Length; //道路长度

Channel; //道路的通道数

RoadDir; //道路的方向

CarWaitList; //从该道路上路的车辆集合

方法：

runCarInInitList //初始化在该道路上路的车辆集合

runCarToRoad //让车辆从车库到路上行驶

getCarInList(road,channel) //获取该条道路上的车辆集合

driveCarInRoad //让每辆在该道路上的车依次移动并标记状态

object-cross

属性:

Id; //路口Id

RoadId; //连接道路Id，依次为上、右，下，左

方法：

getCarInCross //获取与cross相连接的道路上所有的车辆集合

markDir //标记要出路口的车的方向，直行，左转或右转

isConflict //判断当前出路口的车与其他道路要出路口的车是否冲突

moveToNextRoad //出路口到下一条道路

driveCrossRoad // 该路口的车辆出路口

**一个时间片的操作**

for each road:

—For each channel in each road:

getCarInList(road,channel)

driveCarInRoad(road,channel) //每条道路的每个通道上的车向前行驶

for each road:

runCarToRoad //让车辆上路

for each cross:

getCarInCross

markDir

while has car not is end state: //该路口还有车没有达到终止状态

for road in each Cross:

while road has car not is end state;//该条路还有车未转弯

if isConflict:

moveToNextRoad

driveCarInRoad

runCarToRoad

else: //如果冲突则进行下一条道路转弯

break