Intelligens Elosztott rendszerek

(BMEVIMIAC02)

Dokumentáció

Kövér Márton - W6HYOZ
Barton Péter - T2C4UM

2018. tavasz

Feladat leírása

Jelen feladatunkban egy intelligens kereszteződést valósítottunk meg, melyben személyautók, gyalogosok, forgalmi lámpák közös működését próbáltuk meg szemléltetni. A rendszer minden esetben elkerüli a baleseteket, ez volt számunkra a legfontosabb kritérium. Továbbá fontosnak tartottuk még, hogy a megkülönböztető jelzéssel közlekedő autók szabadon közlekedhessenek, és ilyenkor minden más jármű álljon meg.

Egy kereszteződés felülnézetét ábrázoljuk a programunkban, és az ebbe torkolló útszakaszokon közlekedő autók mozgását szabályoztuk különböző kritériumok szerint. Egy útszakasz két sávos, melyen mindkét irányban zajlik a közlekedés, és minden szakaszt egy forgalmi lámpa szabályoz. Amennyiben a lámpák kikapcsolt állapotban vannak (sárgák), az autóknak a jobbkézszabályt kell követni, míg működő lámpáknál értelemszerűen a lámpa színének megfelelő akció az elvárt. A kereszteződésbe bizonyos idő után rendszeresen behajt egy rendőr, ez a többi ágensnek jelez, és amíg nem ér a célállomásához, addig minden járműnek meg kell állnia, hogy szabad utat biztosítsanak neki. (Egy sávban két autó is elfér, ha az egyik nem mozog.)

Egy gyalogos is nehezíti az autók haladását, az egyik úttesten sétál oda, vissza egy kis késleltetéssel, azt is kritériumként kezeltük, hogy a gyalogost nem üthetik el az autók.

A feladat specifikációját a korábban beadotthoz képest minimálisan módosítottuk, mert akkor még nem voltunk tisztában a felhasználható eszközökkel, így egyes feladatok nagyon nehezek lettek volna.

A rendszert egy GridWorldModelben valósítottuk meg, az ehhez tartozó GridWorldView felel az ágensek megjelenítéséért. A rendőr elindulását, illetve a lámpák kikapcsolását beleírtuk a kódba, hogy mikor történjen, és kellő időt hagytunk, hogy lehessen látni, ahogy a rendszer alkalmazkodik az adott szituációhoz.

Megoldás

A rendszer az alábbi ágensekkel rendelkezik:

- Pedestrian
- Cars (car_1, car_2, car_3)
- Traffic lights (lamp_1, lamp_2, lamp_3, lamp_4)
- Police

Az ágensek funkcionalitása:

● Pedestrian

A forgalmat bonyolítva járkál az egyik úttesten át oda és vissza, mozgása során nincs tekintettel senkire, megállás nélkül megy. Az úttesten három lépésből ér át, mielőtt lelépne hosszabb időt várakozik.

● Cars

A forgalom magját képző autók megfelelő közlekedése a feladat legfontosabb része. Minden autó mozgása eltérő, egy-egy útvonalat járnak be. Figyelembe veszik a lámpák jelzéseit, annak hiányában pedig a jobbkéz szabályt követve közlekednek. Ha az előttük lévő mezőn már van másik autó vagy gyalogos igyekeznek elkerülni az ütközést, azzal, hogy megállnak. Ha a rendőr elindul, minden autó azonnal megáll.

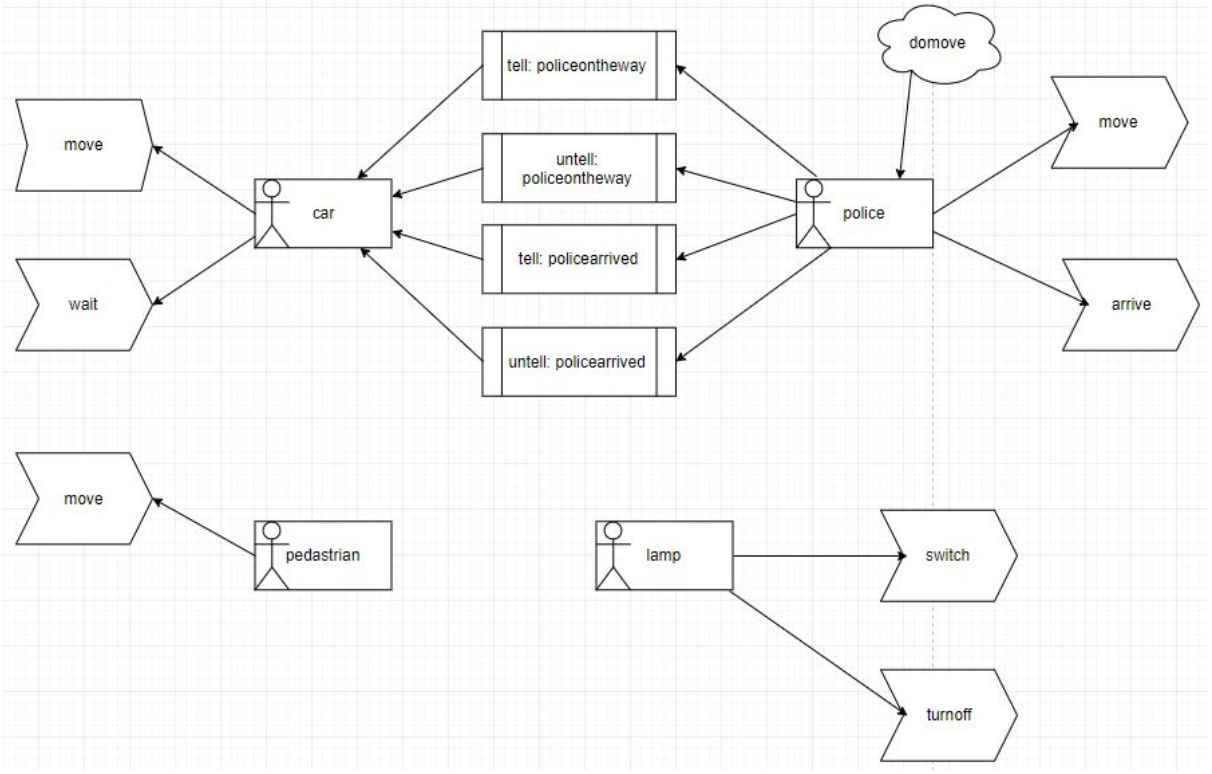
● Traffic lights

A lámpák esetében megkülönböztetjük azt, hogy az adott lámpa üzemel-e, vagy ki van kapcsolva, továbbá ha működésben van, akkor milyen színűen világít. Feladatuk a jármű forgalom irányítása.

- Police

  A rendőr adott időközönként megkülönböztető jelzését használva megjelenik a forgalomban, ilyenkor jelzi minden autónak közeledését, így azok meg tudnak állni helyet biztosítva az elhaladásához. A lámpák működésére nincsen tekintettel.

A rendszer összefoglaló ábrája



Fejlesztés

Jason verzióból a legfrissebbet használtuk. (2.2)

Az információ szerzést nehézkesnek találtuk eleinte, az interneten próbáltunk tájékozódni és ismerkedni a nyelvvel. Az első hetekben csak ez az ismerkedési fázis történt, és a tényleges fejlesztést csak ez után kezdtük el, hogy megfelelően tudjunk haladni. Sokat segített, hogy találtunk példákat GridWorldModel-hez, és sikerült megismernünk a működését.

Java szinten építettük fel az egész felületet, merre a GridWorldModel-t használtuk. Úgy gondoltuk, hogy ez nagyon jól alkalmazható a feladatra, mert pont egy négyzetrácsos hálót képzeltünk el, melyen a négyzetek egy-egy útparcellát ábrázolnak. Megjelenítésre a hozzá tartozó GridWorldView-t használtuk, segítségével jól ki tudtuk rajzolni az ágensek helyzetét, habár vannak hátrányai is, például a villogás, vagy az hogy az egy mezőre lépő ágensek közül eltűnik az egyik és adott mezőn már nem is kerül újrarajzolásra, csak ha odébb lép.

Az autók, a gyalogos és a rendőr mozgását ASL szinten ütemeztük, de javaban valósítottuk meg. GridWorldModelben csináltuk, mert ott a többi ágens pozíciója lekérdezhető, és ez jól szimulálja a különböző szenzorok működését. (Például távolságérzékelő, radar.)

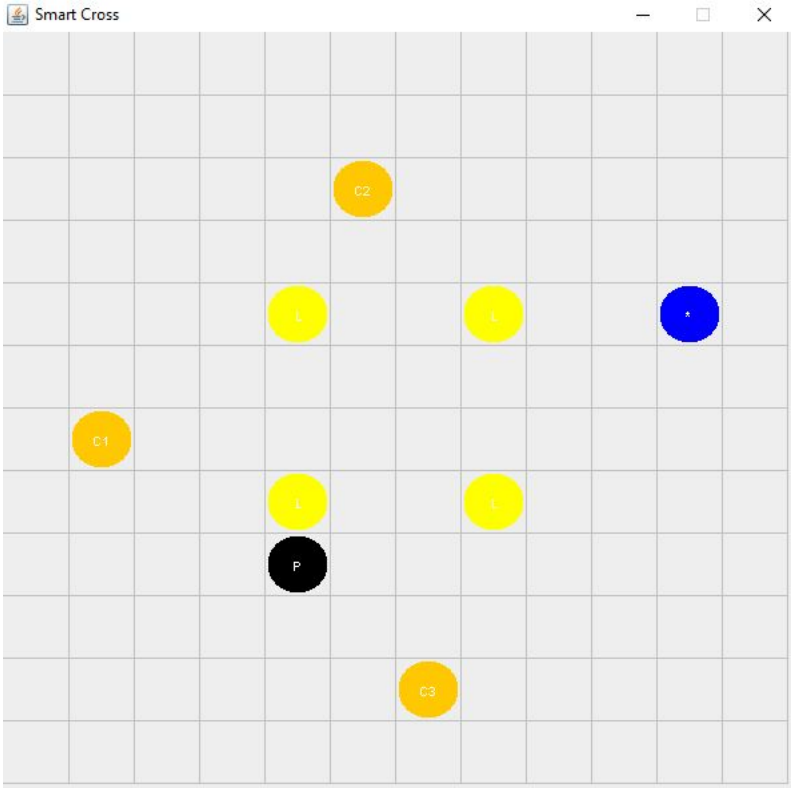
A lámpák müködését ASL szinten valósítottuk meg, azonban állapotukat a java környezetben nyilvántartjuk.

A mas2j fájl foglalja össze az ASL-ben megírt ágenseket és a javaban megírt filet. Az enviroment-ből származó osztály feladata az aslben megírt ágensek és a Javaban megírt modell összekötése egymással.

A kifejlesztett program

Az elkészült program működését bemutató videó itt található.

Programunk egy olyan szimulációt ábrázol, melybe a felhasználó nem tud beleavatkozni, csak szemlélni az eseményeket. Kezdetben a lámpák kikapcsolásra kerülnek és úgy kell boldogulnia a forgalomnak. Bekapcsolásuk után rövid idő múlva a rendőr is elindul. Eztán a rendőr rendszeresen járkál. Az autók és a gyalogos egy megadott útvonalon járkálnak körbe-körbe.



A fenti ábrán a kék csillaggal jelölt kör a rendőrt jelöli, a C és egy szám felirattal ellátottak az autókat, a fekete P feliratú pedig a gyalogost. A képen a lámpák kikapcsolt állapotban vannak, így azok jelenleg sárga színűek és L felirat van rajtuk. Bekapcsolt állapotban a rajtuk szereplő felírat nem változik, de a színük piros és zöld között váltogat a hozzátartozó értéknek megfelelően.
Ha valamelyik ágens csinál valamit, azt próbáltuk átláthatóan logolni, hogy követhető legyen onnan is, hogy mi is történik.

Egyes ágensprogramok rövid összefoglalása

car_1 ágens

Az autók asl kódja csak annyiban különbözik egymástól, hogy melyik autóhoz tartozó move atomi cselekvést hívjuk meg.

Kezdetben a mozgás az ágens célja.

```
+!move : true <-
      move(car_1);
      !move.
```

Ezt a move(car_1) cselekvés segítségével teszi meg, a célja pedig továbbra is a mozgás marad.
Ha a hiedelmei között szerepel a policeontheway miközben a mozogás a célja, megáll és várakozik.

```
+!move : policeontheway<-
      wait(car_1).
```

Ha felkerül a hiedelmei közé a policearrived újra elkezd mozogni.

```
+policearrived:true<-
      !move.
```

pedestrian ágens

A gyalogos kezdeti célja az első lépés megtétele, ami egy hosszabb várakozásból és a lépés végrehajtásából áll, eztán másodikat szeretne lépni.

```
+!firststep : true <-
      .wait(4000);
      move(pedestrian);
      !secondstep.
```

A második és harmadik lépés nagyon hasonlít az elsőre, a kölünbség a várakozás idejében és a következő célban van csak.

```
+!secondstep : true <-
      .wait(1000);
      move(pedestrian);
      !thirdstep.
+!thirdstep : true <-
      .wait(1000);
      move(pedestrian);
      !firststep.
```

lamp_1 ágens

Az útjelző lámpák működése sem tér el sokban egymástól, így csak az egyiket mutatom be, a lamp_1 és lamp_3 akkor piros, amikor a lamp_2 és lamp-4 zöld.
Kezdetben a lámpa kikapcsol

```
+!start: true<-
      !turnoff.
```

Kikapcsolt állapotban elvégzi a kikapcsolást, majd várakozik mielőtt zöldre váltana.

```
+!turnoff : true <-
      turnoff1;
      .wait(20000);
      !green.
```

Zöld esetén várakozik majd végrehajtja a színváltoztatást és a piros lesz az új célja.

```
+!green : true <- .wait(5000);
      switch1;
        !red.
```

Piros esetén ugyan ez a helyzet, csak zöldre vált.

```
+!red : true <- .wait(5000);
      switch1;
        !green.
```

police ágens

Induláskor hosszabb ideig várakozik, majd közli az autókkal elindulását, beállítja a domove hiedelmet, hogy mozoghasson, majd a mozgás lesz a célja.

```
+!start :true<-
      .wait(30000);
      .send(car_1,tell,policeontheway);
      .send(car_2,tell,policeontheway);
      .send(car_3,tell,policeontheway);
      +domove;
      !move.
```

Mozgás során, amennyiben úgy hiszi mozoghat, lép és újra a mozgás lesz a célja.

```
+!move :domove <-
      move(police);
      !move.
```

Ha nem mozoghat közli, megérkezett és újraindul.

```
+!move: not domove<-
      !restart;
      police_arrived.
```

Ha az arrived hiedelem pozitív lesz szól az autóknak, hogy már nem mozog és megérkezett és a domove hiedelmét negatívra állítja, hogy ne folytassa a mozgást tovább. Az arrived hiedelem beállítását a környezet végzi a mozgástól függően.

```
+arrived: true<-
      .send(car_1,untell,policeontheway);
      .send(car_2,untell,policeontheway);
      .send(car_3,untell,policeontheway);
      .send(car_1,tell,policearrived);
      .send(car_2,tell,policearrived);
      .send(car_3,tell,policearrived);
      -domove.
```

Újraindulás során várakozás után közli az autókkal, hogy nincs megérkezve és elkezdi a mozgását, majd a mozgást tűzi ki céljául.

```
+!restart :true<-
      .wait(10000);
      .send(car_1,untell,policearrived);
      .send(car_2,untell,policearrived);
      .send(car_3,untell,policearrived);
      .send(car_1,tell,policeontheway);
      .send(car_2,tell,policeontheway);
      .send(car_3,tell,policeontheway);
      +domove;
      !move.
```