Keretrendszer intelligens, városi közlekedési rendszerek szimulációjához

1. Motiváció

Ahhoz, hogy érdekes kérdésekkel is foglalkozhassunk az intelligens gépkocsiforgalom és intelligens forgalomirányítás témakörében, szükségünk van egy biztos alapra, egyfajta keretrendszerre. Egy ilyen keretrendszer az alapként használt forgalomszimulátor mély rétegeit elfedi, így könnyebbé teszi az implementációs folyamatokat.

Ezzel a keretrendszerrel szemben támasztunk néhány alapvető kritériumot:

- A keretrendszer biztosítson alapvető üzenetküldő és fogadó mechanizmusokat gépjármű–gépjármű, illetve gépjármű–közlekedési csomópont között.
- A keretrendszer adjon interfészt intelligens csomópontok implementációjához.
- A keretrendszer a lehető legnagyobb mértékben fedje el a szimulátor mélyebb rétegeit.
- A keretrendszer oldja meg az intelligens gépjárművek csoportformálását és sávváltásait. Tegye ezt olyan módon, melyet később esetleg módosíthatunk.

Bár korábban, az Önálló laboratórirum c. tárgy keretein belül már létrehoztam egy hasonló rendszert, most az volna a cél, hogy ezt átgondoltabban, továbbfejleszthetően és nem utolsó sorban hibamentesen ismételjem meg. Ezzel lehetőséget teremtve későbbi vizsgálatokhoz elvégzéséhez is.

2. Simulation of Urban MObility

Kiindulási alapként a Simulation of Urban MObility nevű (röviden SUMO, [1]) forgalomszimulátort használom. Ez az egyes járművek viselkedését egyesével szimulálja, tehát ez egy mikroszimulátor.

A szimulátor felépítése nagyvonalakban úgy képzelhető el, hogy a járművek egy gráf élein mozognak. Ez a gráf írja le az úthálózatot, melynek a csúcsai a kereszteződések. A járművek mozgását két logika befolyásolja. Az egyik az autókövetési modell *(car following modell)*, a másik pedig a sávváltási modell *(lane-change model)*.

3. Rétegmodell

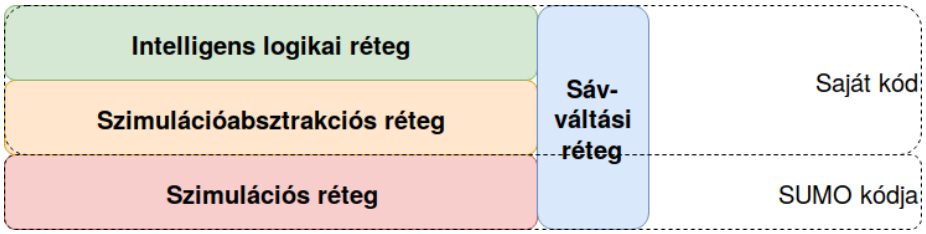
Azért, hogy jól elkülöníthessük a saját logikánkat a szimulátorétól, bevezettem 3 réteget (lásd az 1. ábrát).

A legalsó réteg a *szimulátor réteg*. Ez az a szint, ahol csak olyan kódok szerepelnek, melyek a SUMO forráskódbázisából származnak.

Erre az alsó rétegre épül a *szimulátorabsztrakciós réteg*. Ez egyfajta kapocs a felsőbb rétegek és a szimulációs réteg között. A rétegben lévő kódok ismerik a SUMO kódbázisából származó osztályokat, metódusokat. Viszont ezeket az osztályokat elfedik a keretrendszer felsőbb szintjei elől.

A keretrendszer legfelsőbb rétege két részből áll. Egyik részét a keretrendszert használó újabb modulok fogják adni. A másik részét pedig az egyelőre nem módosításra szánt kódok adják. Ezt a szintet összefoglaló néven *intelligens logikai szintnek* nevezem.

Még egy érdekesség, hogy a sávváltásokért felelős szoftverrész egy saját réteget kapott *sávváltási réteg* néven. Erre azért van szükség, mert ez a kódrész átível a másik három rétegen is, a szimulátor mélyén is dolgozik, hogy megvalósíthasson olyan funkciókat, amelyek egyértelműen az intelligens logika szintjére tartoznának.



1. ábra. A rétegmodell szemléltetése

4. Üzenetküldő rendszer

Korábban explicite szerepelt üzenetküldő rendszer a keretrendszer-implementációmban, most azonban úgy döntöttem, hogy az üzenetküldés függvényhívásokon keresztül meg. Univerzalitást pedig a közös, absztrakt alaposztály interfésze ad a rendszernek (pl. közlekedési csomópontok ősosztálya, járművek ősosztálya stb.). Erre főleg azért volt szükség, hogy a program áttekinthetőbb legyen, így ugyanis egy indirekcióval kevesebb szerepel a programkód szintjén.

Ugyanakkor elméleti szinten továbbra is beszélek üzenetekről, ugyanis ez filozófiájában, leírásában igen szemléletes, csupán forráskód szintjén csak implicite jelenik meg.

5. Az intelligens keretrendszer alapfunkciói

5.1. Csoportkezelés

5.1.1. Markerrendszer

A csomópontok belépési pontjain belépési markerek vannak elhelyezve (lásd a 2. ábrán). A belépési markerek jelzik a járművek számára, hogy be kell lépniük egy csoportba.

A csomópontok kilépési pontjánál pedig kilépési markerek vannak, ezek jelzik, a csomópont elhagyását, a csportból való kilépés szükségességét.

5.1.2. Csoportformálás

Amikor a járművek egy csomópont közelébe érnek, akkor az egy pályán mozgó járművek (azok, amelyek egymás után jönnek egy sávban, és továbbhaladási irányuk megegyezik a csomópontban) csoportot formálnak. A csoportformálás akkor kezdődik, amikor egy autó a belépési markerre ér. A csoport elején haladó jármű a csoport vezetője, a csoport további tagjai őt követik. Amikor a csoportvezető a csomóponton történő áthaladás után a kilépési markerre ér, akkor kilép a csoportból, és a csoport vezetését a mögötte haladó csoporttagra adja át.

Ez eltér a korábban alkalmazott módszertől. Az eltérést az indokolja, hogy így sokkal egyszerűbb egy csoport kettéosztását, egy preemptív csomóponti ütemezést megvalósítani.

A megfelelő csoport kiválasztása belépési markerre érésnél a következő algoritmussal történik:

1. A jármű belépési markerre ér.
2. A jármű megkérdezi, hogy az előtte haladó csoport (ha van ilyen) melyik kilépési pont felé halad.
3. Ha megegyzik a haladási irány, akkor rákérdez arra, hogy csatlakozhat-e még a csoporthoz. Nem csatlakozhat akkor, ha a csoport aktuális mérete elérte megengedett legnagyobb csoportméretet.
4. Ha a járművünk csatlakozhat a csoporthoz, akkor meg is teszi ezt, ellenkező esetben (vagy ha nincs előtte máisik jármű) pedig egy új csoportot alapít.

5.1.3. Viselkedési szabályok csoporton belül

A csoportot a csoportvezető irányítja. A csoportvezető dönt sávváltásokról. A csoporttagok követik a csoportvezetőt, illetve az előttük álló csoporttagokat. Csoporttagok nem előzhetik meg sem egy egymást, sem a csoportvezetőt.

Ha a csoportvezető sávot váltott, akkor a csoporttagok a csoport elejétől kezdve egyesével sávot váltanak a csoportvezető által mutatott irányba.

A csoporttagok egy P-szabályozó segítségével próbálnak meg egymástól megfelelő követési távolságot tartani. Ennek a szabályozónak a munkapontja jelenleg 10 m-re van beállítva.

Természetesen jó lehetőség lenne a fogyasztás csökkentésére, ha minimalizálni tudnánk a követési távolságot, és a reakcióidőket kommunikációval javítanánk. Azonban ezzel a kérdéssel jelenleg nem foglalkozunk, ennek az előnyei amúgy is inkább egy gyorsforgalmi úton jelentkeznének főleg (nagyobb sebességénél nagyobb a szélárnyék, ami kedvezően hat a fogyasztásra).

5.2. Sávváltási rendszer

Az okos sávváltás alatt most a következő dolgot érthetjük. Csoportok haladnak párhuzamos közlekedésre alkalmas útszakaszon egymás mellett. Az egyik csoportnak *(egónak)* sávot kell váltania azért, hogy a célja felé haladhasson tovább. Ekkor egy másik csoportot *(altert)* megkér, hogy engedje be őt. Alter pedig lelassít vagy megáll, hogy ezt lehetővé tegye számára.

A kommunikáció alter és egó csoportvezetője között zajlik, az egész csoportjukat ők képviselik. Alter teljesen altruistán viselkedik, amint egó kérést intéz felé, további mérlegelés nélkül végrehajtja egó kérését. Egó viszont megfontolja azt, hogy a beengedési kérése jogos-e alter felé.

5.2.1. Implementáció

Kiindulási alapként a SUMO kódbázisával érkező „SL2015”-ös névre hallgató alsávmodellt is tartalmazó osztályát használtam. Ez az alapkód már tartalmaz logikát annak eldöntésére, hogy éppen sávot kell-e váltani. Abban az esetben, ha igen, azt is meghatározza, hogy a célsávon haladó gépjármű elé vagy mögé érdemes-e váltani, majd ezt a szomszédos járművel le is kommunikálja.

Ehhez a számításhoz saját paraméterei közül csak a járműhosszát használja. Ezt kis módosítással a csoport hosszára változtattam, majd teszteltem, hogy ez milyen befolyással van a program működésére. Bebizonyosodott, hogy alapvetően az elvárásoknak megfelelő hatása van.

Mivel alter – ahogy már szó volt róla – altruista viselkedésű, ezért némiképp módosítani kellett a beengedő algoritmust is, hogy elérjük ezt az önfeláldozó viselkedést.

5.3. Csomóponti rendszer

A csomópontok vagy bírók felelősek azért, hogy a közlekedési csomópontokon az összes jármű biztonságban, és lehetőleg minél gyorsabban átjuthasson. Keretrendszeri szinten ez egy teljesen absztrakt konstrukció, a megvalósítása már a keretrendszertől független, ugyanis kellően messze van már a szimulációs rétegtől. Egy példát láthatunk a 3. ábrán, ahol az AbstractJudge és ConflictClass a keretrendszer által adott osztályok, míg ezek leszármazottjai azok egy-egy konkrét implementációja.

5.3.1. Konfliktusosztályok

Azon autócsoportok, akik egyszerre áthaladhatnak a közlekedési csomóponton, azok egy konfliktusosztályt alkothatnak. Azon autók haladhatnak át egyszerre egy közlekedési csomóponton, akik egymást nem metsző pályán mozognak.

A konfliktusosztályba sorolás implementációfüggő, a keretrendszer a konkrét megvalósítástól független, bár egy alapvető implementációt kényelmi okokból nyújt.

5.3.2. A bíró, mint ütemező

A bíró feladata, hogy a konfliktusosztályok közül kijelölje, hogy ki haladhat át a csomóponton. A konkrét megvalósítás itt is implementációfüggő, de alapvetően az operációs rendszerek ütemezői szolgáltatják az algoritmusok ötletét.

5.3.3. Preempció és megvalósítása

Mint az operációs rendszerek ütemezői, a bírólogika is viselkedhet preemptív módon, azaz megszakíthatja bizonyos konfliktusosztályok áthaladását, még akkor is, ha a konfliktusosztályt képző járműcsoportok még nem teljes egészében haladtak is át.

Hogy ez létre jöhessen, az egyes járműveket külön-külön kezeljük, nem csoportszinten. Az egyes járművek pollingozással, periodikusan lekérdezik a bírót, hogy áthaladhatnak-e a csomóponton. Amennyiben a válasz nemleges, és még nem haladtak át egy olyan ponton, ahol már nem lehet biztonsággal megállni, a járművek lelassítanak, szükség esetén megállnak (jelenleg a csomópont közepétől 25 m-re).

Ez azért hatékony megoldás, mert egy csoportvezető pl. áthaladhat még a csomóponton, a rögtön őt követő csoporttag meg már nem. Ebben az esetben a csoporttag megáll, a csoportvezető pedig eléri a kilépési markert, és ezzel a csoport vezetését már rá is bízta az immár álló csoporttagra.

6. Néhány gyakorlati példa

6.1. Bíróimplementációk

6.1.1. Round Robin alapú bíró

A keretrendszer kipróbálásaként implementáltam egy Round Robin jellegű csomóponti logikát. Ez gyakorlatilag egy klasszikus rendőrlámpa „intelligens” megfelelője.

6.1.2. MDDF alapú ütemező

Operációs rendszerek témaköréből ismerjük, hogy a legrövidebbet előre elnevezésű (Shortest Job First, SJF) ütemező optimális az átlagos várakozási időkre

nézve. A forgalomszimulációban az algoritumus löketidőbecsléseként a célig hátralévő távolságot vesszük, ahogyan azt a [2]-ben olvashatjuk.

Tehát ez az implementáció a következőképpen működik:

1. A mozgáspályáknak megfelelően konfliktusosztályokba soroljuk a járműcsoportokat.

2. Bizonyos időközönként (pl. 3 másodpercenként) meghatározzuk, hogy melyik konfliktusosztályban van az a jármű, amely a legközelebb van a célállomásához.

3. Az előző lépésben megtalált konfliktusosztályt jelöljük következő áthaladónak.

Jelenleg ennek az implementációja még nincsen sajnos kész.

6.2. Jövőbeli felhasználási lehetőségek

6.2.1. Megkülönböztető jelzést használó járművek kezelése

A keretrendszer segítségével lehetőség nyílna megvizsgálni, hogyan lehet javítani a megkülönböztető jelzést használó járművek előrehaladását egy sűrű városi forgalomban.

Ez egy igen fontos kérdés, hiszen a jelenlegi közlekedés egyik nagyon sarkalatos pontja az, hogy a járművezetők hogyan reagálnak, ha egy szirénázó mentő, rendőr, tűzoltó megjelenik az úton. Sajnos a mindennapi tapasztalatunk azt mutatja, hogy ritkán sikerül igazán segíteni az előrejutásukat. Éppen ezért lenne előnyös kidolgozni eljárásokat, módszereket arra, hogy egy intelligens rendszer hathatósan segítse a megkülönböztető jelzést használó járművek haladását.

Itt érdekes kérdés, hogy a rendszer hogyan tudja előre engedni a szirénázó gépjárműveket. Vagy egy intelligens közlekedési csomópont hogyan tudja biztosítani a mielőbbi áthaladását úgy, hogy közben a forgalom többi résztvevője is biztonságban marad.

6.2.2. MI-beli algoritmusok használata forgalomirányításra

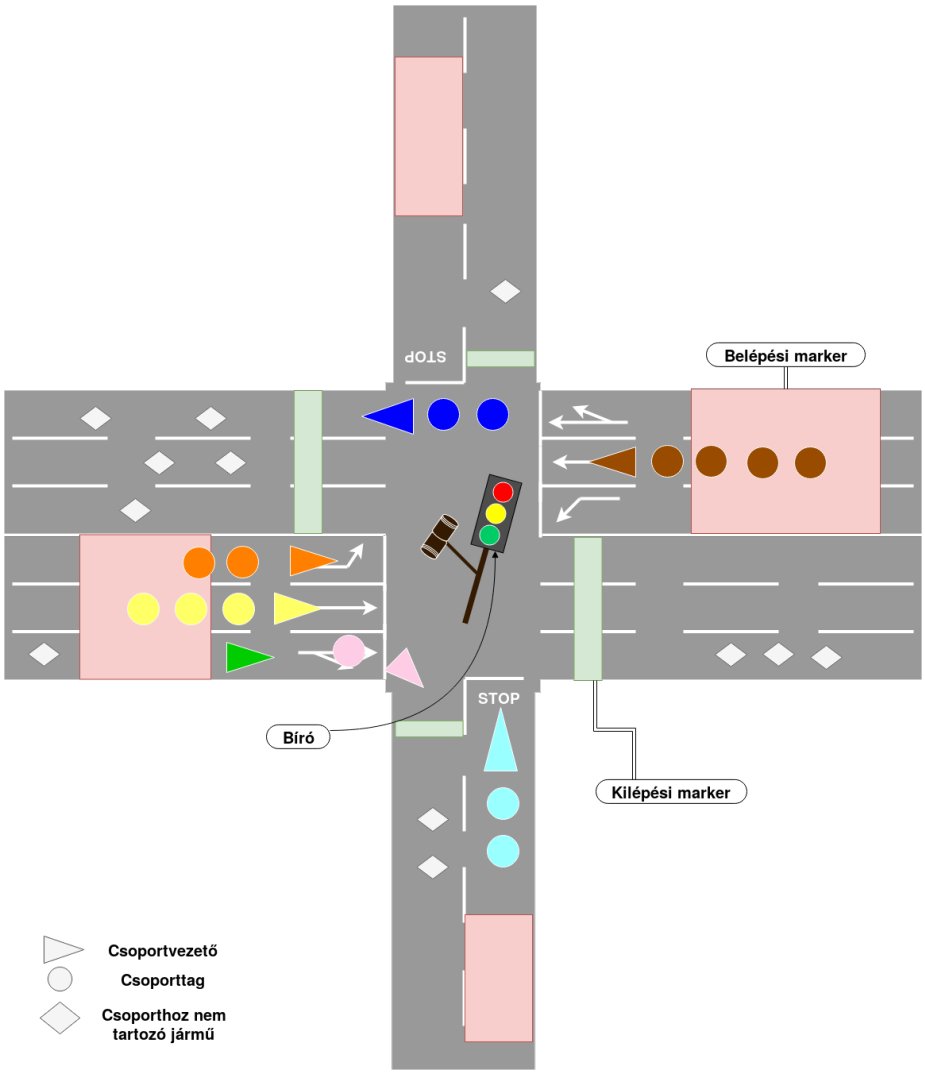
Az absztrakt bírálogika megengedi, hogy bármilyen forgalomirányító algoritmust implementálhassunk. Érdekes kérdés lehet, hogyha nem egy klasszikus megközelítést (Round Robin vagy valamilyen operációs rendszerütemezőt) használunk, hanem valamilyen mesterséges intelligenciából ismert algoritmusra (pl. A*) építünk.

Érdekes lehet, hogy nyerünk-e ezzel valamit a klasszikus rendőrlámpás vagy egy MDDF-alapú megoldáshoz képest. Ugyancsak érdekes, hogy mi történik akkor, ha egy ilyen rendszerben egy megkülönböztető jelzést használó gépjármű szeretne áthaladni a csomóponton.

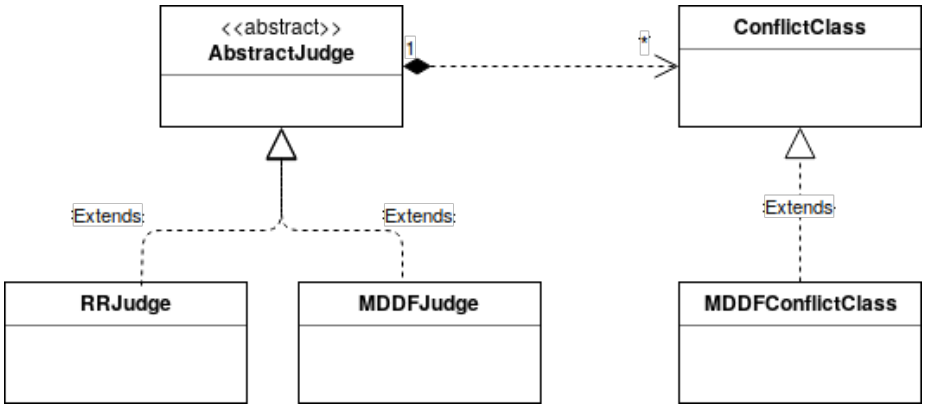
Hivatkozások

[1] D. Krajzewicz, J. Erdmann, M. Behrisch, és L. Bieker, „Recent Development and Applications of SUMO – Simulation of Urban MObility”, o. 11, 2012.

[2] F. Ahmad, S. A. Mahmud, G. M. Khan, és F. Z. Yousaf, „Shortest remaining processing time based schedulers for reduction of traffic congestion”, in 2013 International Conference on Connected Vehicles and Expo (ICCVE), Las Vegas, NV, USA, 2013, o. 271–276.



2. ábra. Példa a komponensekre



3. ábra. Az absztrakt osztályok szemléltetése