Robotfutball

Szabó Richárd

1. A rendszer kialakulása

A robotfutball a kilencvenes évek közepén elinduló és azóta egyre nagyobb népszerűségnek örvendő versenyzési lehetőség. Létezik mind szimulált, mind valóságos formában, az utóbbi további ligákba tagolódik a robotok méretétől függően ([Robocup]).

Első megközelítésben a robotfutball témaköre egy izgalmas játéknak tekinthető. Igaz, hogy a témával foglalkozók a versenyek során lefolytatott mérkőzéseket érdeklődéssel követik, de a feladat megoldása alapvetően egy mérnöki-tudományos munka eredménye.

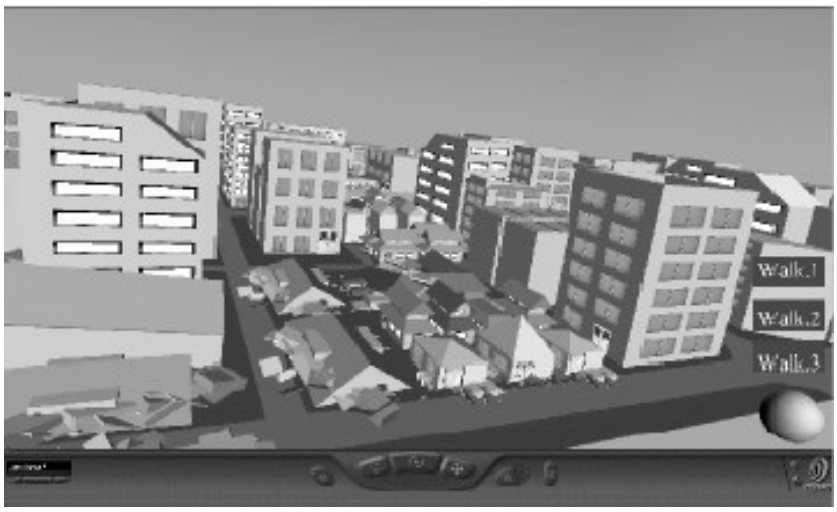
Valódi futball esetén a robotok specifikációnak megfelelő kialakítása és kontrollerjük megvalósítása a feladat. A kísérletező tudóscsapatoknak komoly világcégek nyújtanak segítséget. A kutatómunka nem titkolt célja 2050-re olyan ember formájú és emberien működő robotcsapat létrehozása, amely képes legyőzni a humanoid futball akkori világbajnokát.

Ha lehet, akkor a szimuláció ligája még több érdeklődőt vonz, mivel itt nincs szükség a költséges hardver megtervezésére, " csupán" elméleti kérdéseket kell megoldani, és program formájába önteni. A 2000-es Melbourne-i világbajnokságon már több, mint negyven csapat küzdött az első helyért.

A verseny más területen is gyökeret ver: újabban mentőrobotok is versenyeznek egymással, melyek katasztrófák esetén igyekszenek a további károkat megakadályozni (tűzoltás, sérült villany- és vízvezetékek javítása), és a bajba jutottakat megsegíteni (RoboRescue).

A robotfutball a tudományos kísérletezésen kívül megjelent az oktatásban is, Robocup Junior néven tizenévesek is megismerhetik a robotok működésének alapjait ([Robocup Junior]).

E tudományterület egyik fontos művelője P. Stone, a robotfutball szimulációs szabályainak egyik megalkotója és számos sikeres csapat létrehozója



1. ábra: A RoboRescue (forrás: [RoboCup Rescue])

doktori disszertációját a pittsburghi Carnegie-Mellon Egyetemen ebből a témából és hozzá kapcsolódó kérdésekből írta meg 1997-ben ([Stone 98]). A következő fejezet nagyjából Stone dolgozatát tekinti át, a feladat jellemzői és a szimuláció szabályainak ismertetése után a bajnokcsapat programjának összetevőibe nyújt bepillantást.

2. A robotfutball jellemzői

A szimulált robotfutball egy osztott, multi-ágens rendszer, melyben csapattársak és ellenfelek is szerepelnek. Multi-ágens, hisz több játékos együttes működéséről van szó, ugyanakkor osztott, mivel az irányítás nincs központosítva, azaz minden egyes játékos egy különálló, önállóan futó program.

A játék–kísérlet a játékosok számára nem teljes információs, a korlátos percepciós lehetőségek miatt. A döntéseket tovább nehezíti, hogy mind az érzékelés, mind a cselekvés zajos, a valódi értékektől véletlen eltérések mutatkoznak a valósághűség növelése érdekében.

Az eddig vizsgált szimulációs környezetektől eltérően nincsen meg a hagyományos érzékelés-döntés-cselekvés ciklus, hanem a bemenet és a kimenet aszinkron módon kapcsolódik egymáshoz. Ez annyit jelent, hogy a robotnak memóriájában tárolnia kell a környezetről összegyűjtött információkat, majd

cselekvés idején felhasználhatja azokat.

A több egyszerre működő játékosból adódóan célszerű valamilyen kommunikációt folytatni közöttük, hogy a világról alkotott képüket szinkronizálni lehessen, illetve, hogy az alkalmazandó stratégiát egyeztethessék. Ennek érdekében a játékosok időközönként beszélhetnek egymással. Ez a lehetőség korlátos annyira, hogy a játékosok ne tudják teljesen szinkronba hozni tudásukat, így a feladat teljes információs játékká nem alakítható. Ebből következően a játékos számára csapattársai és ellenfelei bizonytalan kimenetelű cselekvéseket végeznek, vagyis mozgásukat legfeljebb jósolni lehet.

A nehézségeket tetézi, hogy az egész feladatot valós időben kell megoldani, ezért a számításokat véges időn belül el kell végezni.

A valódi robotfutball ettől némiképp eltérő feladat megoldását követeli meg. Az absztrakt érzékelés helyett valódi képek feldolgozásával kell törődni. Nem a robotokra szerelt kamerák, hanem a játéktér fölött elhelyezett központi kamera képe nyújtja a vizuális információforrást mindenkinek. Ezáltal az érzékelés nem teljesen osztott és rejtett állapotról sem beszélhetünk, mint ahogy az előbbi esetben. A kommunikáció teljes hiánya is a különbségeket növeli.

A két feladat sok hasonlóságot hordoz, ezért a feladat megoldását képviselő elmélet és a létrehozott programkód mindkét területen felhasználható. Ugyanakkor az eltérések jelentősek annyira, hogy az eredmények nem ültethetőek át egy az egyben egyik területről a másikra.

Érdekes módon az elmélet a futballtól olyan egészen távoli területen is alkalmazható, mint hálózati csomagok útvonalkeresése. A feladat maga lényegesen eltérő, de olyan tulajdonságok, mint osztott érzékelés és cselekvés, csapattársak jelenléte, rejtett állapot, valós idejűség mind a hasonlóságot növelik, ezért bizonyos alapelvek itt is hasznosíthatóak.

3. A szimulátor felépítése

A következő fejezet a robotfutball szimulátorának 4. változatát tekinti át. A terület népszerűsödésével egyre több új ötlet és változtatási igény jelentkezett a játékkal kapcsolatban. Ma egy bizottság dönt az újdonságokról, melyeket az adott évi világbajnokság után szavaznak meg és implementálnak az új változatban. 2000. őszén már a 6. változat volt az aktuális. A levelezési listán a Melbourne-i világbajnokság óta (2000. szeptember) folyamatosan tárgyalják az érintettek a továbblépés lehetőségét.

A program logikai felépítése az évek során viszonylag keveset változott: kliens-szerver architektúra, melyben a központi program, mint szerver hangolja össze a kliens-játékosok működését. A kliensek a szabványos UDP kapcsolaton keresztül kommunikálnak a szerverrel, egy LISP-szerű nyelven adatokat kapva és utasításokat küldve.

A játékhoz tartozik egy monitor program is, ami a korábban lefolytatott és elmentett mérkőzéseket képes visszajátszani.

A játék a valódi futballnak megfelelő: 22 kliens program küzd a labda ellenfél hálójába juttatásáért. A szabályok sem térnek el lényegesen, melyek teljesülését a bíró biztosítja (bedobás, gól, les, befejezés).

A változtatható környezet érdekében a szerver nagyon sok paramétert tartalmaz, gyakorlatilag a következőkben felsorolt jellegzetességek mindegyike egy vagy több paraméter értékén nyugszik. Ez arra készteti a csapatok fejlesztőit, hogy programjuk alkalmazkodóképes legyen a környezet változásaihoz.

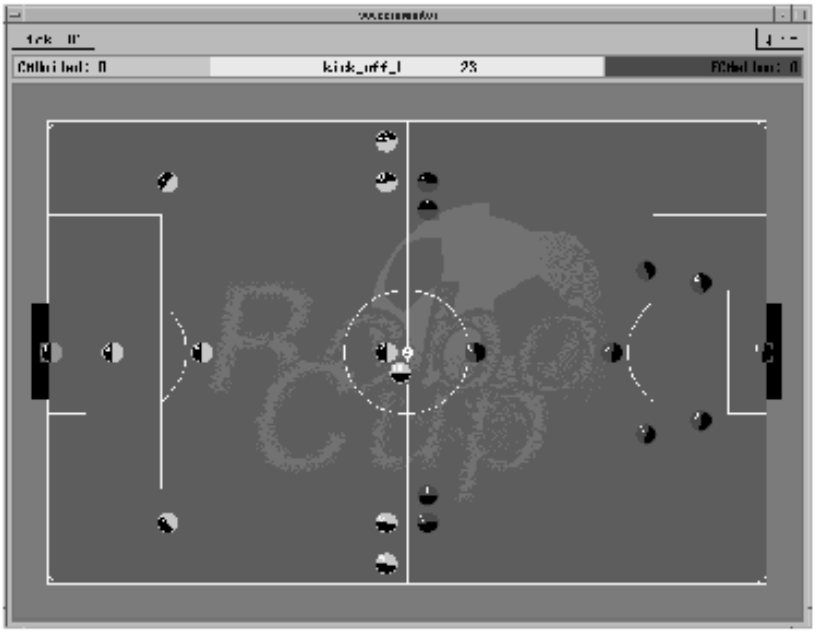
A 2. ábrán a szimulátor képe látható. A játékosokat kettős félkörök reprezentálják, világos oldaluk az első felük. A sötét téglalapok a kapukat jelzik. A múló idő az ablak felső sorában követhető.

Az érzékelés és cselekvés között jelentkező aszinkronitáson túl sem a játékosok egymással, sem egy játékos a szerverrel nem működik összhangban. Ez újabb nehézségeket jelent egy csapat létrehozójának.

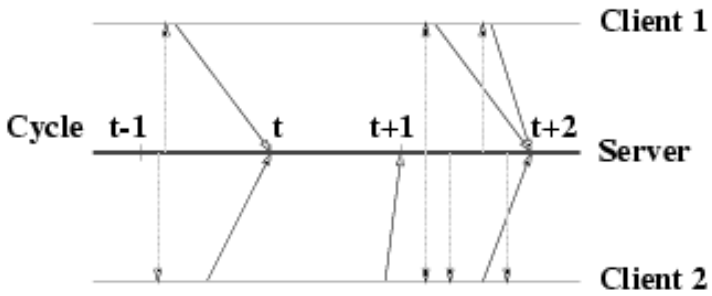
A szerver meghatározott időközönként léptet egyet a belső órán. Két időpillanat között a játékoshoz elküldi az érzeteket, illetve a kiadott cselekvő utasításokra reagál. Egy időszeletben egy játékos csak egy cselekvést végezhet, így több küldött utasítás közül a szerver egyet választ ki véletlenszerűen végrehajtásra. Egy ilyen eset a játékos szempontjából meglehetősen szerencsétlen, hisz nem tudhatja melyik parancsának fog engedelmeskedni a környezet. A kevés utasítás küldésének így van értelme, viszont arra is érdemes vigyázni, hogy tétlen időszeletek sem maradjanak. A környezet állapotának lekérdezésével is óvatosan kell bánni, mert az is a drága időszeletből vesz el.

A 3. ábra a szerver és két játékos működését követi néhány pillanaton keresztül. A 2. játékosnak az érzetek szórt beérkezése ellenére is sikerült minden időszeletre egy cselekvést időzítenie, míg az 1. játékos hol tétlen maradt, hol túl sok utasítást küldött egyszerre.

A szimulált világ egy kétdimenziós futballpálya, melynek lényegesebb pontjain, illetve az oldalvonal mentén kis virtuális zászlók vannak elhelyezve, hogy a játékosok tájékozódását elősegítsék. Zászlók jelzik a kezdőrúgás helyét, a kapu közepét és a kapufákat, a szögletzászlót és a büntető sarkait is.



2. ábra: A szimulátor (forrás: [Stone 98])



3. ábra: A szimuláció lépései (forrás: [Stone 98])

A labda és a játékosok ebben a környezetben vannak elhelyezve, meghatározott x és y koordinátákon. A tárgyak jellemzője még irányuk és sebességük is. Az ezen adatokból kialakuló állapottér igen jelentős, legalább 10^{198} lehetséges állapotot tartalmaz az alapbeállítások mellett.

A tárgyak mozgását egyenletrendszer írja le, mely a pozícióra, az aktuális sebességre és a játékosok által kifejtett erőből következő gyorsulásra épül. A labda mozgását a játékosokkal való ütközés, illetve a rúgó utasítások befolyásolják. Ütközéskor a szimulátor az átlapolásokat igyekszik elkerülni, ezért visszaszámol addig a pozícióig, amíg a két tárgy nincs átfedésben, és a sebességeket invertálja. Nem lehet figyelmen kívül hagyni, hogy a mozgásokat mindig egy zaj is kíséri, ami növeli a valósághűséget.

Annak érdekében, hogy a játékosok ne tudjanak állandóan maximális sebességgel futni mindegyiküknek van egy energiaszintje és a cselekvéseknek egy hatásfoka. Az energia túlzott csökkenését visszafogott mozgással folyamatosan kompenzálni lehet, a regenerálódási ráta keretein belül.

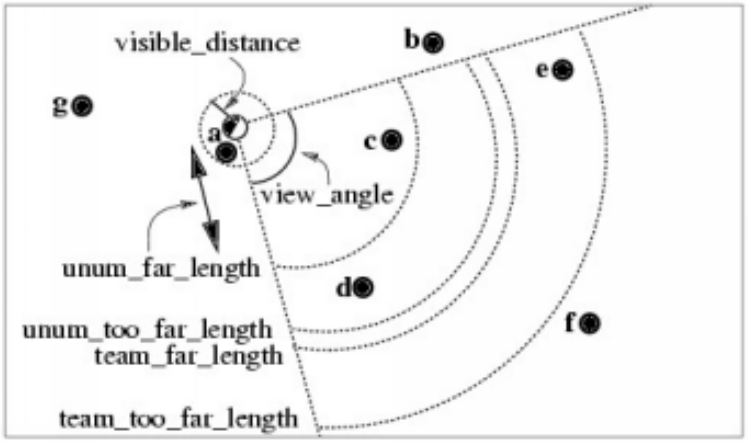
3.1. Az érzékelés

A játékosok érzetei három különböző tartományból érkeznek: hallás, látás, fizikai jellemzők megismerése.

Az audio információk egy megbízhatatlan, alacsony áteresztőképességű csatornán érkeznek a huszonkét játékoshoz a szervertől. A hallás igazából `(hear Time Direction Message)` formátumú üzeneteket jelent, melyben `Time` a szerver aktuális idejét, `Direction` az üzenet beérkezésének irányát és `Message` magát az üzenetet tartalmazza. Látható, hogy az üzenet küldője nincs explicit módon feltüntetve.

Ráadásul az üzenetek csak egy bizonyos távolságon belül hallhatók, egyedül a bíró üzenetei jutnak el mindenkihez. Az alacsony áteresztőképesség annyit tesz, hogy összesen egy üzenet továbbítódik minden két időszeletben, vagyis az ezen felüli üzenetek elvesznek. Ezekből adódóan a hallás meglehetősen megbízhatatlan, vagyis erre az információforrásra kevéssé szabad építeni.

A látás az előbbinél bőségesebb adatokkal szolgál, ezért hasznosabb. A játékoshoz `(see Time ObjInfo ObjInfo ...)` formában jutnak el a látható tárgyak. Minden objektum leírása név, távolság, irány, távolság- és irányváltozás adatokat tartalmaz, a két utolsóból lehet a sebességre következtetni. egy objektum lehet játékos, kapu, labda, zászló vagy oldalvonal is.



4. ábra: A látás jellegzetességei (forrás: [Stone 98])

Minden játékos maga dönthet arról, hogy mennyire részletes információkat kapjon. A látószög például lehet széles, normális vagy keskeny. Bár nyilván egy széles látószög több információt hordoz, mivel negyedannyira sűrűn érkezik a játékoshoz, ezért valamikor érdemesebb a gyengébb módok közül választani.

A kép felbontása is változtatható abban az értelemben, hogy az irány és a távolság adatok helyett a játékos kérheti csak az irány információk elküldését. Az előzőhöz hasonlóan itt is ritkább frissítés járul az összetettebb adatokhoz.

A már többször említett zaj a látást sem kerüli el. Ennek egy érdekes jellemzője, hogy mértéke függ a tárgyak távolságától. Vagyis távolabbi tárgyak távolságáról pontatlanabb információ érkezik, mint a közeliekről.

A látás a halláshoz hasonlóan korlátos környezetben működik. A pontosság azonban koncentrikus köröknek megfelelően változik. A közeli tárgyakat a játékos pontosan látja, más játékosok esetén azok csapata és sorszáma is látható. Ezen kívül a sorszám már egyre csökkenő valószínűséggel érzékelhető, és csak a hovatartozás marad meg. Egy bizonyos távolság után a csapat beazonosítása is egyre kevésbé valószínű, végül csak annyi derül ki, hogy egy másik játékos van az adott irányban. A 4. ábra ezt a mechanizmust mutatja be.

A játékos által megszerezhető fizikai információk kategóriája a játékos energiájáról, teljesítményéről, újratöltődéséről ad tájékoztatást, megadja az aktuális sebességet és a látás fent vázolt paramétereit. Ezek az adatok a

játékos kérésére érkeznek válaszul mindenféle korlátozás nélkül. A sebességgel kapcsolatban megjegyzendő, hogy csupán a nagyságát lehet megtudni, az irányt a látvány változásából lehet leszűrni.

3.2. A cselekvés

A játékosok által végrehajtható cselekvések szintén három csoportra oszthatóak: a kommunikáció, a mozgás és a fizikai jellemzők állítását végző utasításokra.

A hallásnál leírt kommunikációt a játékos `say` utasítással kezdeményezheti. Ebben egy 512 bájt hosszú ASCII sztringet tud elküldeni, amit mindkét csapat játékosai meghallhatnak.

A környezet manipulálására négy utasítás kínálkozik: a `turn`, a `dash`, a `kick` és a `catch`. A szerver ezen utasítások közül mindig csak egyet hajt végre egy időszelet végén. Ha több utasítás is érkezik, akkor ezek közül nem determinisztikus módon választ. Ezért a játékosnak kell tudnia eldönteni, hogy sikerült-e az adott műveletet végrehajtani.

A mozgási utasítások mindegyike szög és/vagy erő értékkel paraméterezett. Fordulás esetén 180^o-ot lehet megtenni. A szögsebesség függ a sebességtől, azaz minél gyorsabban halad a játékos annál kisebb szögben képes fordulni.

A `dash` nevű futási utasítással a játékosok az aktuális haladási irányba mozoghatnak, előre körülbelül háromszor gyorsabban, mint hátra. A folyamatos futáshoz a parancsot minden ciklusban ki kell adni, különben a játékos megáll. A szerver későbbi változataiban már megjelent a nyak, amivel a látvány és a haladási irány nem feltétlenül esik egybe.

A `kick` utasítással lehet a labda rúgását elvégezni. Az utasítás paramétere a rúgás ereje és iránya, amerre a labda gyorsítása bekövetkezik. A rúgásnak négy jellemző tulajdonsága van:

- A labda mozgásának megváltoztatása nem közvetlen beállítással, hanem vektorösszeadással történik.
- A labdát a `kickable_area` nevű szerver paraméter nagyságának megfelelő területen belül tudja a játékos eltalálni Ez a valóságban persze nem így működik.
- A labda és a játékos ugyanúgy ütközhet, mint más tárgyak.

- A rúgás ereje a labda és a játékos egymáshoz viszonyított helyzetétől is függ: a legnagyobb akkor lehet, ha labda közel van a játékos előtt. Ha a labda távol van a `kickable_area`-n belül, akkor a rúgás gyengébb lesz. Hasonlóan, a játékos haladási irányától való nagyobb eltérés csökkenti a rúgás erejét.

Védést (`catch`) csak a kapusok végezhetnek, ők is csak a büntetőterületen belül. A védés csak akkor hatékony, ha a labda a `kickable_area`-hoz hasonló `catchable_area`-n belül van.

A fizikai jellemzők változtatása a játékos paramétereinek lekérdezését (`sense_body` utasítás), illetve a látás tulajdonságainak megváltoztatását jelenti (`change_view` utasítás).

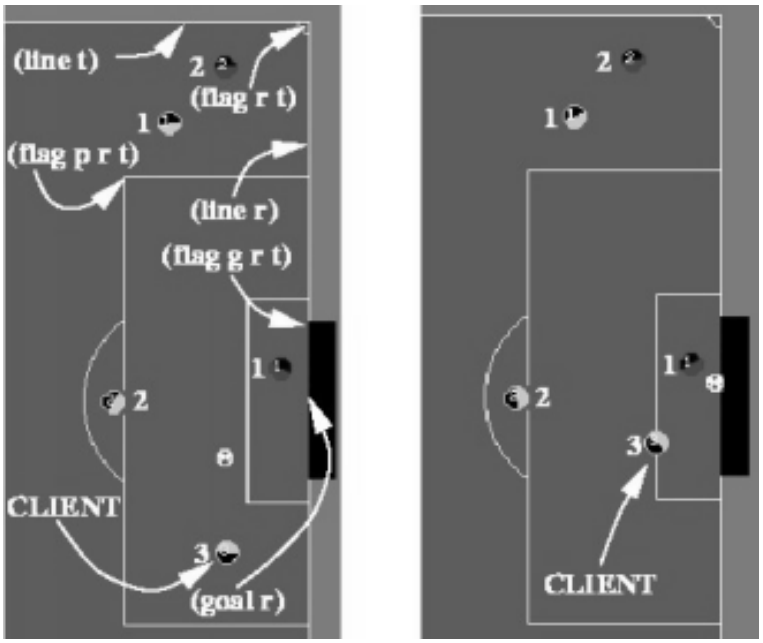
3.3. Egy jellegzetes eseménysor

Az eddigiek összefoglalásául egy játékos körülbelül két másodpercig tartó mozgását követi nyomon a 5. ábra. A megjelölt sárga mezes játékos a bal oldalon látható pozícióból a jobb oldaliba jut el miközben a labdát a kapura rúgja.

A 1. táblázat kódrészlete a közben zajló adatcserét jeleníti meg a játékos szempontjából. Követhető, hogy az idő 94-től 113-ig változik. Látszik az is, ahogy az érzetek az idő folyamán változnak, mire a játékos a kapura fordul a csapattársak már nem látszanak. A listából látható, hogy a két távolabbi játékos sorszáma nem érzékelhető ilyen messzeségből. Az időről időre kiadott `sense_body` utasítások folyamatosan tájékoztatnak a játékos állapotáról. A utolsó sorok a bíró üzeneteit tartalmazzák: jelzi a gólt és a középkezdést.

4. A CMUnited-98 felépítése

Egy szimulált robotfutball-csapat létrehozása a szerver összetettségének ismeretében nem egyszerű feladat. Sokféle feladatot kell megoldani, melyek absztrakciós szintjükben is lényegesen eltérnek egymástól. A szerverhez tartozó egyszerű példaprogram csak alacsonyszintű fájlműveleteket végez az UDP kapcsolaton keresztül. A csapatrészek viselkedésének szinkronizálása ettől meglehetősen távol van. Az eddigi meggondolások alapján látszik, hogy a csapatot irányító programot érdemes hierarchikusan elkészíteni, az eltérő absztrakciós szinteket szétválasztva. A CMUnited-98 is ennek megfelelően készült el.



5. ábra: A játékos mozgása (forrás: [Stone 98])

A tervezéskor az egyik legfontosabb dolog, amit figyelembe kell venni, hogy a futball csapatjáték, ezért már a kezdeti tervezésnél a játékosok közötti együttműködést kell szem előtt tartani. A valós idejűség és a limitált kommunikációs lehetőségek miatt nem lehet komplex egyeztető protokollokat alkalmazni. Ezért a CMUnited-98 a Periodic Team Synchronization-t alkalmazza, azaz időszakonként a csapat viselkedését egyeztetik a robotok. A játék közbeni korlátos kommunikáció időszakait a kezdeti, a félidei és időkéréskori szabad egyeztetés lehetősége váltja fel, amikor minden játékos tudását felhasználva lehet kialakítani a stratégiát. A korlátos kommunikáció nehézségeit másrészről a locker-room agreement oldja meg, ami annyit tesz, hogy minden játékos az előírtaknak megfelelően viselkedik és ezt feltételezi csapattársairól is.

Egy játékos általános felépítése ezek figyelembe vételével alakul ki (6. ábra).

Az architektúrát alkotó hét modul közül az interpreter gyűjti be a külvilággal kapcsolatos információkat és a világról alkotott képhez teszi hozzá. A világ állapota a robot számára meghatározott valószínűséggel rendelkező

```
**-> (dash 100.00)
(see 94 ((goal r) 15.3 27) ((flag r t) 47.9 8) ((flag p r t) 34.8 -15) ((flag p r c) 16.4 -34 0 0)
    ((flag g r t) 21.8 19) ((ball) 8.2 0 0 0) ((player CMUnited) 40.4 -8)
    ((player CMUnited 2) 16.4 -37 0 0 117) ((player Opponent 1) 16.4 15 0 0 -148)
    ((player Opponent) 44.7 0) ((line t) 47.5 89))
**-> (sense_body)
(sense_body 95 (view_mode high normal) (stamina 1280 1) (effort 1.0) (recovery 1.0) (speed 0.39)
*->  (dash 100.00)
(see 96 ((goal r) 13.6 31) ((flag r t) 46.1 8) ((flag p r t) 33.1 -16) ((flag p r c) 14.9 -39 -0.298 -0.9)
    ((flag g r t) 19.9 20 -0.398 0.5) ((ball) 6.7 -2 -0.402 0) ((player CMUnited) 36.6 -8)
    ((player CMUnited 2) 14.9 -41 -0.298 -0.9 117) ((player Opponent 1) 14.9 17 -0.298 -148)
    ((player Opponent) 40.4 0) ((line t) 45.6 89))
**->  (dash 100.00)
**-> (sense_body)
(sense_body 97 (view_mode high normal) (stamina 1120 1) (effort 1.0) (recovery 1.0) (speed 0.44)
**->  (dash 100.00)
.
.
.
(hear 103 -70 shoot the ball) (see 104 ...((ball) 1.8 6 0.108 5.4) ...)
**-> (say shooting now)
**-> (kick 100.00 65.40)
(hear 104 self shooting now)
**-> (sense_body)
(sense_body 105 (view_mode high normal) (stamina 980 1) (effort 1.0) (recovery 1.0) (speed 0)
**-> (turn 31.76)
(see 106 ... ((ball) 4.1 14 1.23 7) ...)
**-> (turn 14.00)
.
.
.
**->  (dash 100.00)
(see 112 ((goal r) 6.8 12) ((flag r t) 38.5 -32) ((flag g r t) 12.3 -14 -0.246 0) ((ball) 7.4 2 0.74 1.5)
    ((player Opponent 1) 7.4 -18 -0.148 -0.2 107) ((player Opponent) 33.1 -44) ((line r) 8.2 -40))
(hear 113 referee goal_l_1)
(hear 113 referee kick_off_r)
```

1. Táblázat: A játékos kommunikációja a szerverrel (forrás: [Stone 98])

hitek gyűjteménye.

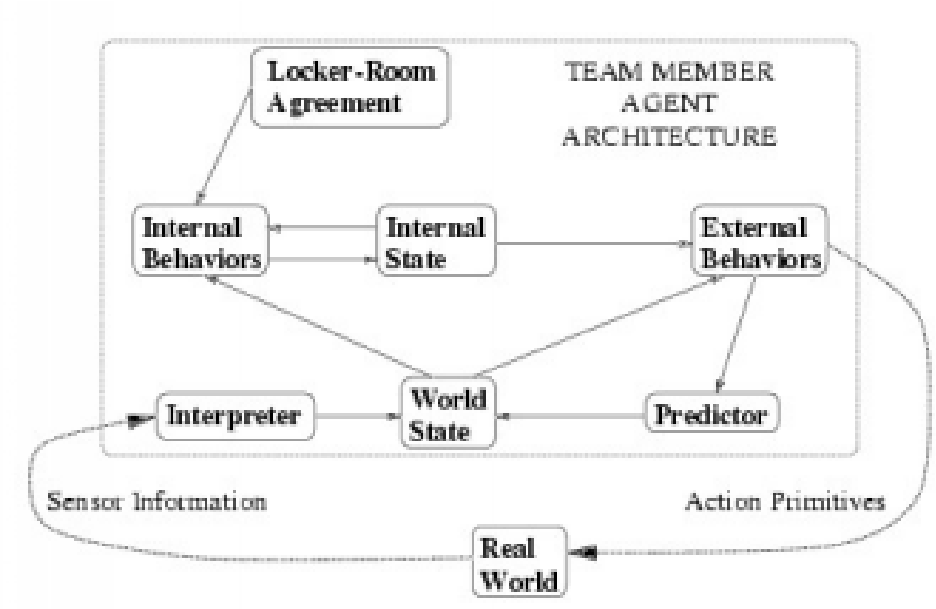
Az állapotok másik halmaza a belső állapotoké. Ezek a játékos belső változóit tartalmazzák, meghatározva a csapatbeli helyet és a szándékokat.

A már említett locker-room agreement a flexibilis csapatmunkát meghatározó megállapodás, ami például tartalmazza a kommunikációs protokoll leírását vagy a lehetséges csapatformációkat.

A belső viselkedések közül a belső állapotok, a világ állapota és a locker-room agreement alapján választ a játékos, célja a belső állapotok összhangba hozása az érzékelt világgal.

A hasonló felépítésű külső viselkedések kiválasztását ezzel szemben a belső állapotok és a világ állapota befolyásolja. A robot külső viselkedésével elsősorban a külvilágot alakítja, de a prediktoron keresztül, ami a viselkedés által kiváltott hatást jósolja, a játékos világról alkotott képét is megváltoztatja.

Már volt arról szó, hogy a feladat sokrétűsége miatt célszerű hierarchikus struktúrát kialakítani a robotjátékosok irányítására. Az előbb bemutatott moduláris felépítés is ebbe az irányba tett lépés, azonban a viselkedéseket



6. ábra: Egy játékos felépítése (forrás: [Stone 98])

a belső és külső csoportoknál tovább is lehet tagolni. A CMUnited-98 viselkedései fastruktúrát alkotnak, melynek legfelső szintjén egy belső és egy külső viselkedés áll, melyeket időközönként végrehajt a játékos. A fa szintjei a különböző absztrakciós szinteknek felelnek meg, melyeket a tanulás szempontjából mutat be a 7. ábra.

A játékosok felépítésének másik igen fontos eleme a csapatjátékot támogató tulajdonságok. Annak érdekében, hogy a csapat hatékonyan, megbízhatóan és mégis változatosan viselkedhessen Stone szerepeket, formációkat és begyakorolt figurákat definiál.

A szerepek olyan belső és külső viselkedésekre értelmezett specifikációk, melyek tetszőleges absztrakciós szintén meghatározhatók. Ilyen lehet például egy középpályási szerepkör. A formáció szerepek halmaza, mellyel a csapaton belüli együttműködést lehet meghatározni. Egy formáció egy csapategységet tartalmazhat egy kapitánnyal. A szerepek és formációk nem fix struktúrái a játékosoknak és a csapatnak, menet közben cserélődhetnek, változhatnak. Végül a begyakorolt figurák szerepek halmazára értelmezett feladatsorok, egy indító feltétellel, ami a környezet állapotából következik. Az egyes szerepekhez a figurában viselkedésleírás és terminálási feltétel tartozik, aminek

7. ábra: A viselkedés szintjei (forrás: [Stone 98])

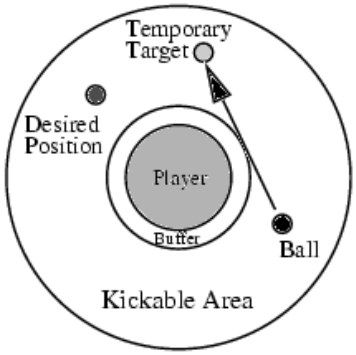
végrehajtása után a játékos visszatérhet normális működéséhez.

A csapatmunka másik része a kevéssé megbízható, de mégis fontos kommunikáció megvalósítása. Jól megválasztott üzenetekkel természetesen mindenféle feladat megvalósítható, de ehhez a kommunikációnak bizonyos feltételeknek kell megfelelnie. A legfontosabbak, hogy az üzenetküldő játékosnak azonosíthatónak kell lennie, védekezni kell az ellenfél esetleg szándékos, zavaró üzenetei ellen, a küldött üzenetnek rövidnek és hibatűrőnek kell lennie.

A fenti feltételeknek megfelelő üzenetformátum a locker-room agreement része. A játékosok és a formációk egyedi sorszámot kapnak, így egyértelműen azonosíthatóak. Az ellenfél hamis üzenetei ellen kódolt időbélyegző véd. Egy véletlen üzenetben a hibás formátum miatt egyszerű az elutasítás. Ha az ellenfél újraküldi a saját üzenetünket, azaz az üzenet formátuma megfelelő, akkor az időkésés miatt következik be elutasítás. A hibatűrés képessége abból adódik, hogy a játékosok csak segítségnek használják a kommunikációt, de nem függenek tőle.

Érdekes kérdés, hogy amikor egy küldött üzenetre választ vár a játékos, akár több társától is, akkor miként lehet megkerülni a két ciklusonként egy üzenet korlátját. Ilyen esetben minden játékos a saját sorszámának megfelelő kivárással küldi el válaszát, így az üzenetek között nem lesz ütközés.

A CMUnited-98 esetében a kommunikáció legfontosabb feladata a formációk szinkronizálása. Ehhez a játékosok felhasználják a kódolt időbélyegzőt,



8. ábra: A labda elrúgása (forrás: [Stone 98])

és mindig a legkésőbbi üzenet veszik figyelembe. Ha kapnak egy üzenetet, amely régebbi, mint a legutolsó állapotváltásról szóló, akkor automatikusan helyesbítő választ küldenek a tájékozatlan felé.

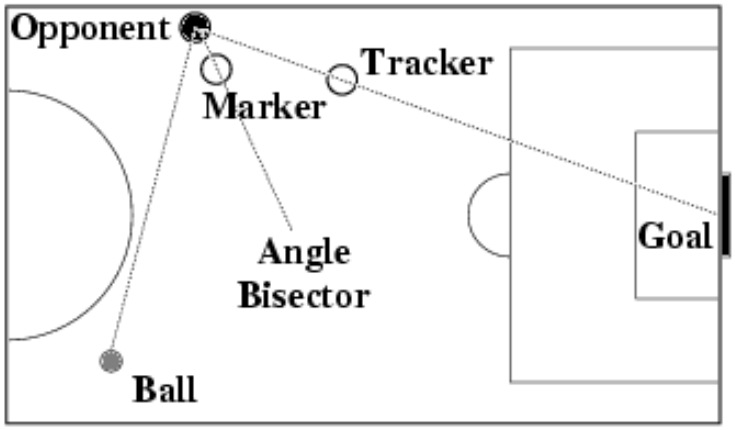
5. Elemi viselkedések

A csapat részletes felépítésének leírása túlságosan terjedelmes, ezért csupán az előre huzalozott alacsonyszintű viselkedésekből szeretnék bemutatni néhányat. Ezek mind egy játékos elemi műveletei valósítják, azaz lokális hatásúak.

Az egyik alapvető viselkedés, nem túl meglepő módon a labda megrúgása, melyre a játékos körül a `kickable_area` területén van lehetőség. Mivel a labda tetszőleges irányból érkezhet és tetszőleges irányba kell tudni továbbítani, ezért nem egyszerűen egy `kick` parancs kiadásáról van szó. Szükség esetén a játékos érintője mentén kell a labdát továbbítani az átellenes oldalra (8. ábra).

A rúgás elvégzésének másik kérdése a célsebesség és az irány meghatározása. Természetesen eltérő sebesség szükséges lövés és passz esetén. A `kickable_area` definíciójából adódóan a játékos megteheti, hogy több kis rúgással a labdát felgyorsítja, és így jobb eredményt ér el, mint egy nagy rúgással.

A labda vezetését végző elemi viselkedés felhasználja a rúgás elemeit. Ez a funkció egyszerűen a játékos haladásának és a labda rúgásának felváltva végrehajtott sorozata. Ennek a két lépésnek összhangban kell lennie, még-



9. ábra: A védekezési módok (forrás: [Stone 98])

pedig az aktuális és a jövőbeni pozíciók kiszámítása alapján. Az igazán okos labdavezetés ezen felül még az akadályokat és az ellenfeleket is figyelembe veszi haladás közben, az ellenfél játékosainak pozíciója mindig úgy befolyásolja a labda helyét, hogy a játékos testével fedezhesse azt.

A labdakezelés eltérő viselkedést jelent a mezőnyjátékosok és a kapus számára. Az előbbiek a labda mozgásából jósolják jövőbeni pozícióját és megpróbálnak eljutni a legközelebbi elérhető helyre. Ott vagy elakad a labda a játékosban vagy egy kis rúgás ellensúlyozza a labda sebességét. Kapus esetén elég a labdához közel kerülni és a `catch` parancsot kiadni.

A védekezésre két módszert is alkalmaznak a CMUnited-98 játékosai (9. ábra). Az egyik mód a blokkolás, amikor a védekező az ellenfél és kapu közötti vonalat állja el. A másik esetben a játékos a kapu, az ellenfél és a labda által bezárt szög felezőjében tartózkodik, így nehezítve meg a passzolást és a kapura rúgást is. A védekező mindkét formánál figyeli és követi az ellenfelet.

Végül a felszabadítás elemi viselkedés célja a labda eljuttatása az ellenfél térfelére. Ennél a cél az ellenfél játékosainak elkerülése, és a csapattársak megtalálása.

A további, magasabb absztrakciós szintet képviselő viselkedések megvalósítása rétegelt tanulással történik. Ez annyit jelent, hogy egy szint megfelelő képességűvé fejlesztése után a következő szint teljesen ráépül erre a szintre, bemenetét tőle kapva, kimenetét neki küldve.

A legalsó szintű tanult viselkedés egyezik egy korábbi, előre definiált elemi viselkedéssel, a labdakezeléssel. A kettős implementáció a szerver két különböző változatához készült. Segítségükkel összevethető volt a két megközelítéshez szükséges fejlesztési munka, ami nagyjából összemérhető. A tanulásos megközelítésnél a szerző többrétegű neurális hálót alkalmazott.

Erre a szintre épül a passz kiértékelésének modulja, ami azt határozza meg, hogy melyik játékoshoz milyen valószínűséggel sikerül eljuttatni a labdát, a két játékos távolsága és az ellenfelek elhelyezkedése alapján. Az értékfüggvény játékról játékra egyezik, ezért egyszeri tanulással és a megoldást a viselkedésbe építve megfelelően lehet eljárni. A passzlehetőségek egyedi kiértékelése a környezet 174 folytonos attribútumára alapozva történik meg, egy hagyományos döntési fát építve a híres, itt nem részletezett C4.5 algoritmus szerint.

Miután a passzok sikerének a valószínűsége kalkulálható, el kell dönteni, hogy ténylegesen ki kapja meg a labdát. A passz kiválasztása a környezet állapota és a passzkiértékelő függvény alapján végezhető el. Mivel egy döntés eredménye csak a csapat egész meccsen nyújtott teljesítménye alapján értékelhető, ezért a megerősítéses tanulás Q-learning nevű algoritmusának egy módosított változatát (TPOT-RL) alkalmazta a szerző, ami folyamatosan követi a cselekvéseket és később pontozza őket.

Ugyan a szerző a hierarchia tervezésében ennél is továbbmegy, az itt vázolt csapat elég komoly volt ahhoz, hogy többszörös bajnok legyen, és a későbbi években is inspirációt és alapanyagot adjon a többi versenyzőnek.

6. Köszönetnyilvánítás

Ezen cikk és a robotfutballt is tartalmazó jegyzetem ([Szabó 00]) megírásához ösztönzést és segítséget témavezetőmtől, Kampis Györgytől kaptam, amiért hálás vagyok neki. Az elkészítésben nyújtott segítségéért szintén köszönet illeti Salamon Andrást, aki hasznos megjegyzéseivel járult hozzá az írás elkészüléséhez.

Hivatkozások

[Robocup] *Robocup*, URL: http://www.robocup.org

[Robocup Junior] *RoboCup Junior*,
URL: http://www.robocup.org/junior/index.html

[RoboCup Rescue] *RoboCup Rescue*,
URL: http://robomec.cs.kobe-u.ac.jp/robocup-rescue/

[Stone 98] P. Stone: *Layered learning in multi-agent systems*, PhD thesis, CMU-CS-98-187, 1998

[Szabó 00] Szabó R.: *Mobil robotok szimulációja*, Egyetemi jegyzet, 2000